# Bouwen van webpagina's met HTML5

Deze cursusmap blijft eigendom van 5HART-IT, het bij deze cursus behorende naslagwerk kunt u na afloop van de cursus meenemen.

**Wij verzoeken u vriendelijk om niet in de cursusmap te schrijven.**



**Versie 1.0, maart-2021**

# Inhoudsopgave

# Inhoudsopgave

# Inhoudsopgave

## Inhoudsopgave

# 1 Inleiding

## 1.1 Inleiding

De doelstelling van deze cursus is het leren bouwen van een webpagina met behulp van HTML. Na afloop van de cursus zal je in staat zijn om zelfstandig een statische webpagina te bouwen.

HTML is de essentiële basis voor het bouwen van een webpagina. In de praktijk wordt deze taal bijna altijd in één zin genoemd met de termen JavaScript en Cascading Style Sheets (CSS). Dit zijn twee talen die gebruikt worden voor het toevoegen van stijlen, dynamiek en interactie op webpagina's. Deze termen zullen in de cursus benoemd worden, maar zijn té omvangrijk om in detail te behandelen.
Daarvoor verwijzen wij je graag naar de bijbehorende cursussen:

- JavaScript
- CSS
-

Er zullen bij enkele hoofdstukken tevens een aantal sub-doelstellingen geformuleerd zijn. Deze zijn gelijk aan de vereisten voor het HTML deel van de Microsoft HTML5, CSS3 en JavaScript certificering. Doe daar je voordeel mee.

Voor meer informatie over certificering, kijk dan op:
*https://www.microsoft.com/en-us/learning/exam-70-480.aspx.*

Wij willen je vragen alle tussentijdse oefeningen uit te voeren die voorbij komen. De voorbeelden mag je uit uitvoeren. Aan het eind van ieder hoofdstuk vragen wij je ook een aantal eindopdrachten te maken die het hoofdstuk afsluiten. Het leren werken met HTML doen wij vooral door het zelfstandig uitvoeren van deze opdrachten.

### 1.1.1 Legenda

Leerdoelen voor een hoofdstuk. Hiervan ga je straks de vruchten plukken! Wat kan je allemaal na dit hoofdstuk? Deze leerdoelen matchen met de leerdoelen van het HTML 5 deel van het Microsoft web development examen.

Dit is een code voorbeeld, een blueprint. Deze mag je uitvoeren. Kijk of je het hele voorbeeld begrijpt.

Zelf aan de slag! Geef kleur aan je eigen oefening! Dit is waar het om draait in deze cursus, praktische ervaring opdoen.

**Let op!**
Dit is belangrijk en kan je eventueel veel tijd en zorgen besparen in de praktijk!

Expert tip. Wil je de expert worden in dit vak gebied? Neem deze tips dan ter harte.

# 1 Inleiding

Denk hier eens over na. Dit is vaak een stukje tekst of voorbeeld ter overweging.

Performance indicator. Dit onderdeel beschrijft impact op de performance van je webpagina.

### 1.1.2 Vereisten

Werken aan een webpagina kan vanaf praktisch ieder device. Dit komt mede doordat we voor het maken van een webpagina met HTML niets meer nodig hebben dan een eenvoudige tekst editor en browser. In deze cursus ga je gebruik maken van de tekst editor Notepad++ en web browser Google Chrome.

Als je bovengenoemde pakketten niet op je device hebt staan, of indien je het niet zeker weet, dan kan je deze downloaden door naar onderstaande links te navigeren:

Chrome: https://www.google.nl/chrome/
Notepad++ https://notepad-plus-plus.org/download/v6.9.2.html
Of als Alternatief voor Notepad++ een online editoer: https://www.editpad.org/

Voor het maken en bewerken van oefeningen en opdrachten zullen we gebruik maken van de online codebewerking omgeving genaamd CodePen.io. Wij raden u sterk aan om via https://codepen.io/accounts/signup/user/free een account aan te maken zodat u al onze voorbeelden ook in uw eigen account kunt opslaan.

### 1.1.3 Handige websites

De web ontwikkeling branche is continu in ontwikkeling. Ook browsers proberen hier, waar mogelijk, in mee te gaan.
Het is bijna onvermijdelijk dat er verschillen tussen browsers ontstaan wat betreft ondersteuning voor web development talen (denk aan HTML, JavaScript en CSS) en de features daarvan. Vereisten
Werken aan een webpagina kan vanaf praktisch ieder device. Dit komt mede doordat we voor het maken van een webpagina met HTML niets meer nodig hebben dan een eenvoudige tekst editor en browser. In deze cursus ga je gebruik maken van de tekst editor Notepad++ en web browser Google Chrome.

Als je bovengenoemde pakketten niet op je device hebt staan, of indien je het niet zeker weet, dan kan je deze downloaden door naar onderstaande links te navigeren:

- Google Chrome : https://www.google.nl/chrome/browser
- Notepad++ tekst editor https://notepad-plus-plus.org/download/v6.9.2.html
- Device onafhankelijke tekst editor http://www.editpad.org/

Voor het maken en bewerken van oefeningen en opdrachten zullen we gebruik maken van de online codebewerking omgeving genaamd CodePen.io. Wij raden u sterk aan om via https://codepen.io/ een account aan te maken zodat u al onze voorbeelden ook in uw eigen account kunt opslaan.

Om één van onze voorbeelden straks in uw account op te slaan (ofwel te 'forken'), kunt u bij het betreffende voorbeeld, rechts bovenin op de knop 'fork' klikken:

# 1 Inleiding

Handige websites
De web ontwikkeling branche is continu in ontwikkeling. Ook browsers proberen hier, waar mogelijk, in mee te gaan.
Het is bijna onvermijdelijk dat er verschillen tussen browsers ontstaan wat betreft ondersteuning voor web development talen (denk aan HTML, JavaScript en CSS) en de features daarvan.

Om ons tot hulp te zijn worden er tal van websites ontwikkeld.
Een kleine greep uit een selectie met praktische websites om tijdens en na deze cursus bij de hand te houden:

| | |
|---|---|
| Browser ondersteuning: Can I use it? | http://caniuse.com/ |
| Browser ondersteuning CSS en Javascript: Quirksmode | http://www.quirksmode.org/ |
| Web development oefeningen: W3 schools | https://www.w3schools.com/ |
| HTML en JavaScript naslag: MDN | https://developer.mozilla.org/nl/ |
| Browser prefixes: Peter Beverloo | https://peter.sh/experiments/vendor-prefixed-css-property-overview/ |
| Chrome DevTools: DevTools | https://developer.chrome.com/devtools |

### 1.1.4 Tips voordat u begint

Bij vragen of onduidelijkheid: lees uw code nog eens aandachtig door, let op typfouten, probeer met behulp van Google uw antwoord te vinden, of vraag bij onduidelijkheid of problemen uw docent om hulp.
Indien u besluit voorbeelden of opdrachten te maken, raden wij aan om de meeste van deze opgaven from scratch op te bouwen. Dus zonder teveel code te kopiëren, tenzij u zeker weet dat u het te kopiëren stuk code beheerst.

## 1.2 Wat is HTML

HTML staat voor Hyper Text Markup Language en is de taal waarin webpagina's worden geschreven. Het is een taal die bestaat uit codes die we aan tekst kunnen toevoegen (markup betekent verrijken).

HTML is eenvoudig te leren. Het maken van een webpagina komt kortweg op het volgende neer: we nemen de gewenste tekst op in een tekst-editor en structureren deze met behulp van tags (hierover later meer). Het structureren kan betekenen het aanpassen van lijstjes, maar ook het toevoegen van afbeeldingen, videoclips of geluidsfragmenten. Wanneer het bestand vervolgens wordt opgeslagen met de extensie .htm of .html zijn we in principe klaar.

Een browser zorgt ervoor dat HTML-tekst wordt geïnterpreteerd naar een webpagina. Elke browser (zoals Microsoft Internet Explorer/Edge, Google Chrome, Mozilla Firefox of Opera) kan HTML-tekst verwerken.

# 1 Inleiding

### 1.2.1 Versies

De huidige versie van HTML is HTML 5.0. Hierin zijn een heleboel zaken veranderd sinds het ontstaan van de HTML 4.01 standaard. De focus in deze versie is nóg meer komen liggen op het duidelijk maken van het verschil tussen de structuur en de opmaak van webpagina's. De talen JavaScript en CSS, die in deze cursus regelmatig aangestipt worden, hebben daarmee ook beter een plek gekregen binnen een web development proces.

Enkele belangrijke toevoegingen in HTML 5 zijn onder andere nieuwe typen invoervelden en beter gekozen semantische elementen om structuur van een pagina mee op te bouwen. We zien ook veel nieuwe elementen voor het weergeven van beeld en geluid én het optimaliseren van programma's die in hun geheel online werken (wat een doorzettende trend is).

Let op: in deze cursus gebruiken wij Google Chrome. Omdat er verschil kan zitten in ondersteuning van bepaalde HTML features door verschillende browsers, is het goed om te weten wat handige websites zijn om bij de hand te houden in het geval een element of attribuut niet werkt of een onverwacht resultaat geeft.

Bekijk welke features jouw browser ondersteunt via: https://html5test.com/ of de can I use it webpagina (http://caniuse.com/).

### 1.2.2 W3C

Het World Wide Web Consortium, ook wel W3C, is een bedrijf dat zich richt op het opstellen van regels en richtlijnen bij het gebruik van webtalen zoals HTML, CSS en JavaScript. Binnen deze cursus zullen wij ook meermaals naar het W3C(http://www.w3c.com/) refereren, simpelweg omdat zij de norm bepalen als het gaat om standaarden én validatie(s) daarvan.

### 1.2.3 Structuur

Voordat wij gaan beginnen is het leuk om even te kijken hoe een willekeurige website er achter de schermen uit ziet.

Open de Google Chrome browser (het voorbeeld werkt in een willekeurige browser) en navigeer naar een website naar keuze. Klik vervolgens met de rechter muisknop op: 'paginabron weergeven' of <ctrl><u> in Google Chrome.

We zien nu de inhoud van webpagina zoals deze (grotendeels) in code is geschreven. Schrik niet! Het kán er heel complex uit zien. Dat heeft te maken met het feit dat er meer talen dan enkel HTML gebruikt worden bij de bouw van een webpagina.

Wanneer we kijken naar de opbouw van een HTML webpagina, dient een HTML 5 pagina altijd te beginnen met een zogeheten doctype. We geven daarmee aan aan de webbrowser dat hij deze pagina mag zien als pagina die geschreven is volgens de HTML 5 taal. Voorheen was deze code een stuk uitgebreider er diende een Document Type Definition (DTD) naar keuze opgegeven te worden. Dat is een document wat een stel regels bevat waaraan je document (in ons geval een webpagina) aan moest voldoen. Deze DTD's staat online op de website van het W3C, én staan ook vastgelegd in verschillende website ontwikkeltools voor het geval je even geen Internet verbinding hebt.

Sinds de intreding van HTML 5, mogen we geen DTD's meer gebruiken. Een DTD beperkt op teveel vlakken de mogelijkheden van HTML 5. Sommige zaken zijn namelijk niet te valideren.

```
<!DOCTYPE html>
```

Het grote verschil tussen een moderne HTML 5 doctype declaratie (boven) en een oud HTML 4 doctype (onder).

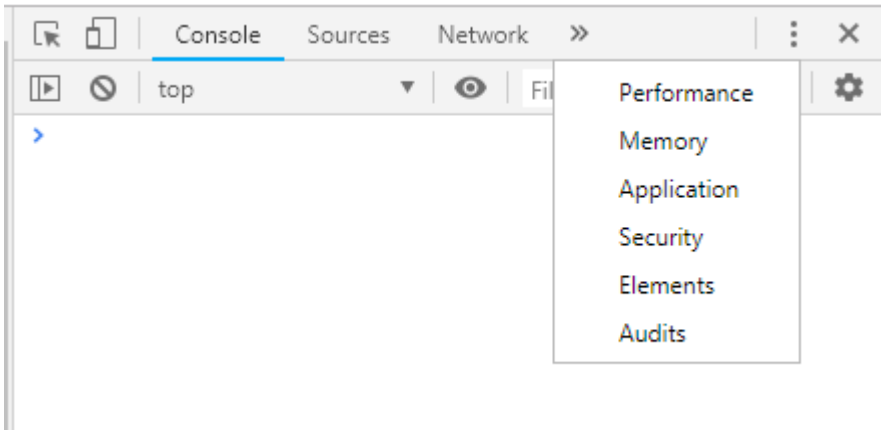
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Wil je de webpagina valideren om te zien of hij correct is opgebouwd qua syntax (schrijfwijze)? Dan kan je dat bijvoorbeeld online doen via de W3C Validator (https://validator.w3.org/). Dit is een tool die aan de hand van een zogeheten XML schema (uitgebreider dan een DTD) én een aantal zelf gebouwde checks controleert of het document voldoet aan de W3C HTML 5 standaarden.

### 1.2.4 Debugging

Fouten opzoeken in je code, ook wel debugging genoemd, was tot een aantal jaar geleden een lastige taak voor web ontwikkelaars. Gelukkig hebben wij tegenwoordig een groeiend scala aan hulpmiddelen tot onze beschikking. Met name voor het opsporen van fouten in JavaScript en CSS code zijn de volgende tools erg prettig om te gebruiken:
'Hulpprogramma's voor ontwikkelaars' in Google Chrome (te starten in de browser middels <ctrl><shift><i>, óf <f12>)
'Ontwikkel hulpprogramma's' in Internet Explorer (te starten in de browser middels <f12>)
Met name de features die het netwerkverkeer (o.a. zogeheten HTTP-requests) monitoren, zijn érg handig om te zien hoe uw website het doet qua performance.

Binnen deze cursus zullen wij de hulpprogramma tools niet gaan gebruiken. Onderstaand wordt nog wel screenshot getoond van hoe deze tools eruit ziet in chrome.



# 1 Inleiding

# 2 Elementen en attributen

## 2.1 Inleiding

In dit hoofdstuk gaan we de basis structuur van een HTML pagina bespreken. We gaan ervaren welke elementen er komen kijken bij het opbouwen van een pagina en hoe je deze elementen uitwerkt als tags in een HTML document.

- Het kennen van de verschillen tussen blok- en inline elementen
- Handmatig HTML elementen kunnen toevoegen en wijzigen
- Semantisch structureren van een pagina met behulp van de tags: section, article, nav, header, footer en aside

## 2.2 Opbouw van een HTML document

Een HTML document, ofwel, een webpagina, bestaat in zijn meest basale vorm uit een aantal elementen. Hiervan is er geen enkele noodzakelijk om als webpagina te kunnen functioneren(!). Er zijn echter wel een aantal sterk aan te bevelen om te gebruiken, alleen al om aan de W3C standaarden te voldoen.

Onderstaand zie je de opbouw van een basale webpagina volgens de W3C HTML 5 standaarden:

```
<!DOCTYPE html>
<HTML>
    <head>
        <title>Pagina titel</title>
    </head>
    <body>
    </body>
</HTML>
```

Elementen (onderdelen) van een webpagina kunnen wij in HTML weergeven met behulp van tags. Een tag begint altijd met een klein haakje: < en sluit weer af met een groter-dan: > haakje. Om een element te maken (bijvoorbeeld een plaatje) typen we de naam van de tag die bij het element hoort, met bovenstaande haakjes er omheen. In het geval van een plaatje dus: <img>.

De meest elementen hebben ook een tag om aan te geven dat het element 'klaar' is. Voor een paragraaf ziet dat er als volgt uit:

Inhoud van de paragraaf

Let op de slash (/) die als afsluiter voor de letter p staat. Alles wat tussen de open en sluit tags staat, is onderdeel van het element.

De elementen die geen tag hebben om af te sluiten, dienen binnen de eerste tag nog afgesloten te worden middels een slash, bijvoorbeeld: <input />. Ook een element dat veel gebruikt wordt om een regelovergang aan te geven, past dit toe: <br/>.

# 2 Elementen en attributen

Het element html, ook wel het root element genoemd, is naast het doctype het eerste element dat genoemd zal worden op een webpagina. Daarmee geef je aan een webpagina te gaan maken, of, los van IT termen: hebben wij het papier gepakt om iets op te gaan schrijven. Misschien is het al opgevallen dat alle andere elementen binnen het HTML element vallen (pas aan het einde van het document wordt de <html> tag afgesloten met </html>.
Er volgt nu een omschrijving van de belangrijkste (en volgens W3C essentiële) elementen van een webpagina:

- <html> Iedere HTML pagina begint na de doctype met de tag <html> en eindigt met de tag </html>. Hiermee wordt het begin en einde van een HTML-document gemarkeerd. De browser zal hierdoor herkennen dat het document HTML-code bevat.
- <head> Hierin staan de titel van de pagina, informatie over de pagina, eventueel stijlen en eventueel JavaScript code (hierover later meer). De header wordt opgenomen tussen de tags <head> en </head>. Informatie die in de header getypt is, zal niet in de browser verschijnen.
- <title> De titel van de pagina wordt opgenomen tussen de tags <title> en </title> en komt in de titelbalk van de browser te staan. Deze tag staat altijd binnen de <head> tag.
- <body> Hierin staat de feitelijke inhoud van de pagina. Dit deel begint altijd met de tag <body> en eindigt altijd met de tag </body>. We zullen later zien dat hierop één uitzondering bestaat.

Wil je zelf een nieuw element maken? Dan kunnen wij dat doen door de bijbehorende (vaak met dezelfde naam) tag in te typen in ons document.

Het is vaak leerzaam bestaande webpagina's te bestuderen qua opbouw.
Om te kijken hoe een pagina in elkaar steekt. Is het aan te raden de broncode van een pagina te bekijken, ook wel de source code genoemd.

Dit doen wij op de volgende manier (verschilt per browser):

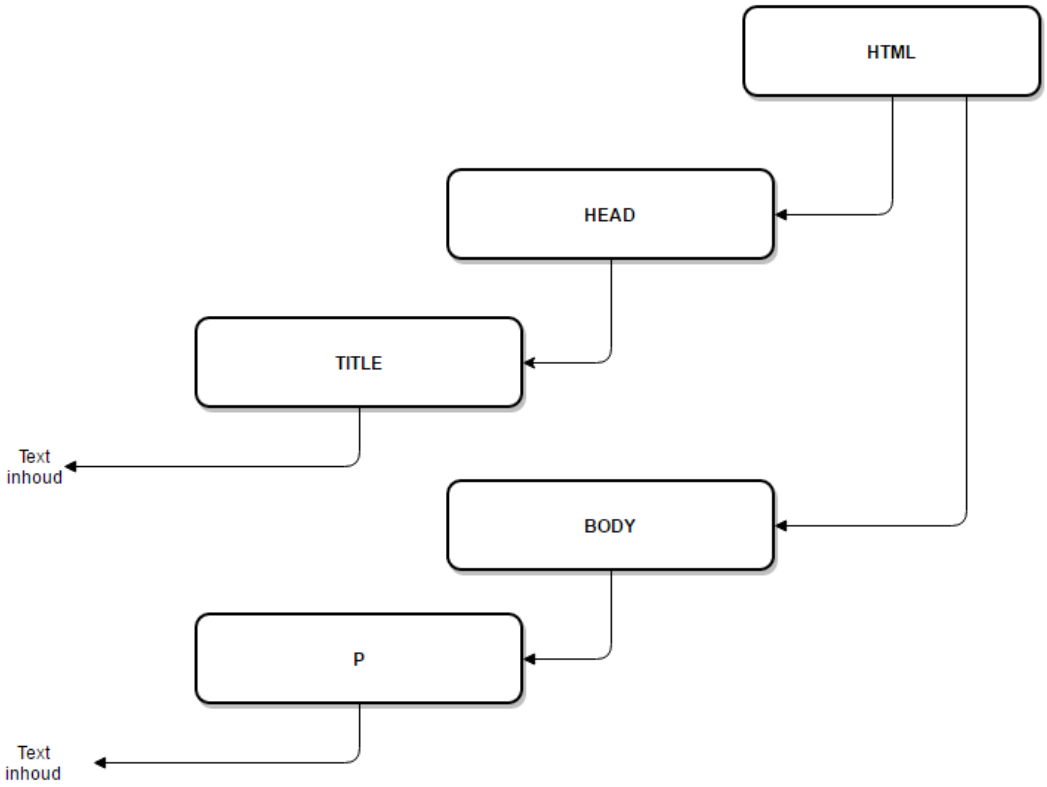
- In Google Chrome: <rechtermuisknop> en kies voor 'paginabron weergeven' of <ctrl><u>.
- In Internet Explorer: <rechtermuisknop> en kies voor 'bron weergeven'.
- In Mozilla Firefox: <rechtermuisknop> en kies voor 'paginabron bekijken'.

De pagina broncode zal vervolgens in een (niet bewerkbaar!) venster verschijnen. De code is uiteraard niet bewerkbaar, omdat éénieder anders zomaar de code van een webpagina zou kunnen aanpassen.

Bekijk de paginabron van een webpagina naar keuze. Probeer daar de meest belangrijke elementen te vinden en te onderscheiden.
Deze oefening is extra, mocht je er niet uitkomen neem dan contact op met de docent.

Een webpagina is schematisch goed weer te geven omdat elementen binnen andere elementen geplaatst mogen worden. Zie hiernaast een schematische weergave van de komende opdracht:

# 2 Elementen en attributen




Maak een nieuw bestand aan met onderstaande inhoud. Noem het bestand: eerstekeer.html.

```
<!DOCTYPE html>
<HTML>
    <head>
        <title>Mijn eerste webpagina</title>
    </head>
    <body>
      Maak van deze tekst een paragraaf
    </body>
</HTML>
```

Zorg er vervolgens voor dat er een paragraaf op het aangegeven punt komt. Een paragraaf wordt gemaakt én afgesloten met een P  tag (dus met het element: <p>).
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

## 2.3 Blok- en inline elementen

Elementen die we in de body van de pagina gebruiken, kunnen grofweg in twee structurele groepen worden verdeeld. De bijbehorende eigenschappen zijn in het algemeen geldig, maar kunnen in specifieke gevallen afwijken.

**Blok elementen**: Deze elementen kunnen andere blok elementen en inline elementen bevatten. Blok elementen vormen grotere structuren dan inline elementen.

Belangrijk: Blok elementen beginnen op een nieuwe regel.

Voorbeelden van blok elementen: <p><form><table><ol><ul><h1>(t/m)<h6>

Een  tag wordt voornamelijk gebruikt voor indelingsdoeleinden. Om een webpagina goed te kunnen beheren, is het praktisch om secties op te delen in hapklare blokken. Deze blokken zijn vervolgens weer als groep te benaderen voor opmaak met CSS code, of voor dynamische content met Javascript.

De <p> tag is er om aan te geven dat een bepaalde tekst als paragraaf geïnterpreteerd mag worden. U zult merken dat veel browsers aan dit element al een bepaalde 'opmaak' hebben gekoppeld, namelijk dat er de nodige wit-ruimte boven en onder dit element wordt getoond in de output.

De elementen table, ol en ul komen in hoofdstuk 5 aan bod. De overige elementen komen in respectievelijk hoofdstuk 3 en hoofdstuk 6 aan bod.

**Inline elementen**: Inline elementen kunnen andere inline elementen en data bevatten. Inline elementen maken deel uit van de regel waar ze worden ingevoerd: ze beginnen dus niet op een nieuwe regel.

Voorbeelden van inline elementen: <a><img />

Een  tag wordt gebruikt om tekst te omvatten. Een span heeft geen standaard opmaak zoals een paragraaf dat heeft. Daarnaast worden twee na elkaar gedefinieerde span elementen gewoon op dezelfde regel getoond (indien de tekst niet te lang is). Bij twee paragrafen zal er altijd een regelovergang aanwezig zijn. Dat is hét merkbare verschil tussen inline en block elementen.

Elementen in het algemeen kunnen ook ingedeeld worden naar wat de inhoud ervan mag zijn. Wij kunnen dan de volgende twee categorieën onderscheiden:

**Lege elementen**: Dit zijn opmaakelementen zoals regeleinden en lijnen. Ze hebben nooit een eindtag en hebben dus geen inhoud, kortom ze bestaan enkel uit één tag.

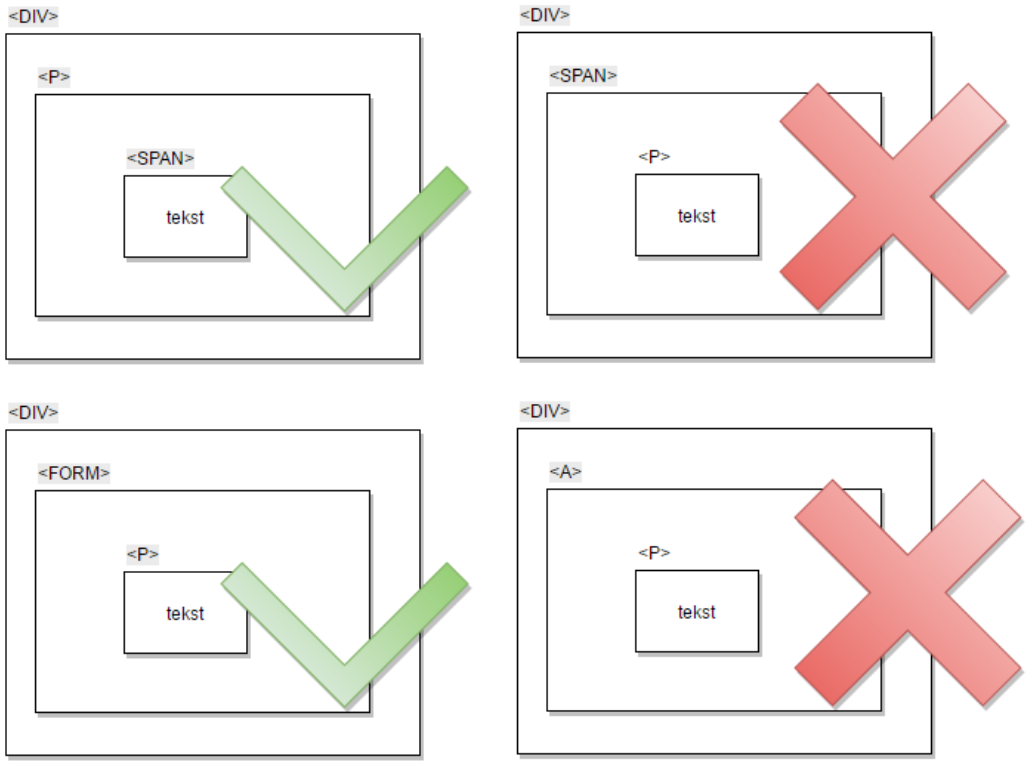
Denk aan: <img /><br />

**Container elementen**: Deze hebben altijd een eind tag. Tussen de tags bevindt zich de inhoud. De inhoud is altijd het gegeven dat moeten worden gestructureerd en vormgegeven (bv: een tekst).

Denk aan: <p>

Er kunnen meerdere tags achter elkaar opgenomen worden (dit heet nesten). Bij de fysieke tekst-opmaakelementen hebben we bijvoorbeeld gezien dat de het gebruik van de b en i elementen, dus met de volgende tags: vet en cursief ervoor zorgt dat een tekst vet én cursief wordt afgedrukt. Let erop dat de tags altijd gepaard gecombineerd

worden. vet en cursief is dus fout, omdat de eindtag  alleen (dus zonder de begintag ) binnen het cursieve tag-paar  en  staat. Een beschrijving van deze benoemde elementen volgt in het komende hoofdstuk.
Onderstaand volgt een schematisch voorbeeld van welke elementen wél, en welke elementen niet binnen elkaar gebruikt mogen worden:




## 2.4 HTML semantische structuur elementen

Bij de introductie van HTML 5 zijn er een zevental elementen bij gekomen die bedoeld zijn om de algemene structuur van een webpagina meer semantische betekenis te geven. Semantisch staat ervoor dat de betekenis (in deze context) van een element als uitgangspunt wordt genomen.

Een vergelijking om dit te verduidelijken kan gemaakt worden tussen: Het b element, wat er enkel voor zorgt dat een tekst dik gedrukt wordt weergegeven en aan de andere kant het strong element wat daadwerkelijk iets zegt over de tekst, namelijk dat het een belangrijke tekst is.

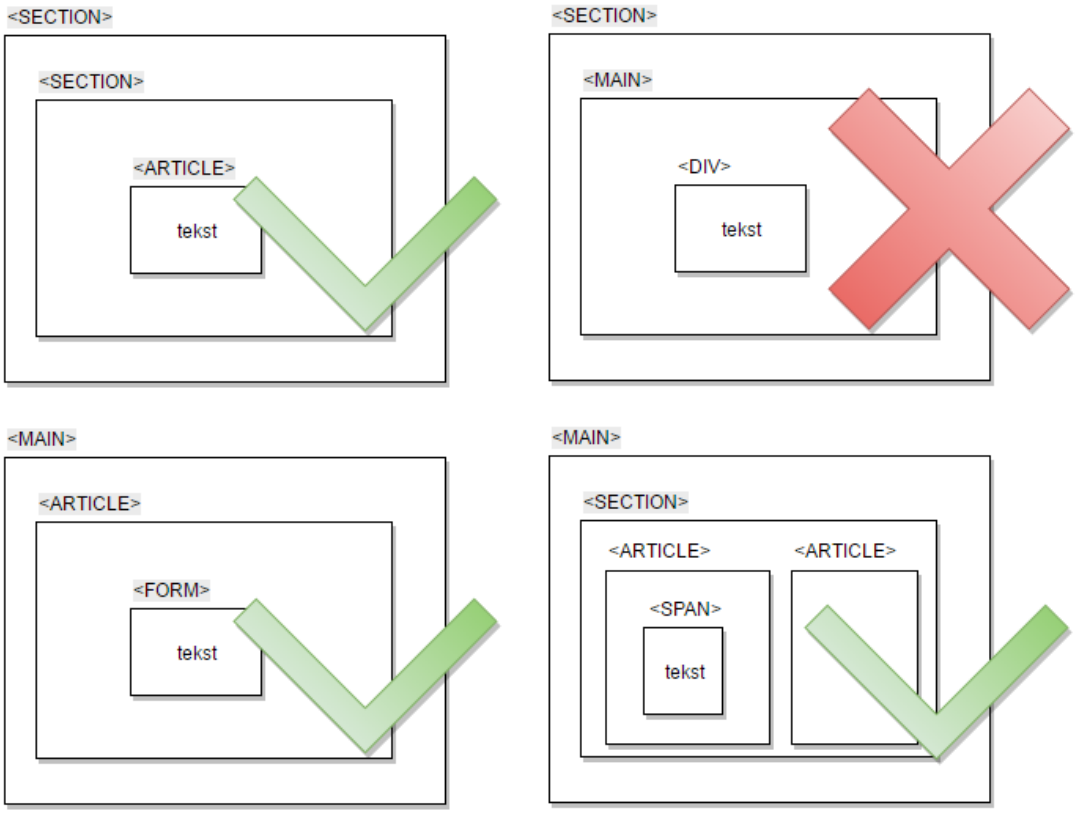
De zeven nieuwe elementen vallen niet onder een categorie block of inline, omdat ze visueel geen effect hebben op de pagina. Ze dienen, zoals gezegd, puur een semantisch doel. Houd in de gaten dat de W3C specificeert dat deze elementen in sommige browsers nog worden voorzien van een eigen opmaak. Hierdoor kunnen bijvoorbeeld regelovergangen of andere stijl eigenschappen opvallen. Lees meer hierover op de W3C website. Een oplossing is om handmatig met CSS code een opmaak toekennen.

# 2 Elementen en attributen

Onderstaand worden deze elementen beschreven:

| | |
|---|---|
| Main | Er zal in de praktijk vaak een  element gemaakt zijn, waar het belangrijkste content deel van de webpagina in komt te staan. Vaak heeft deze zelfs het zogeheten 'id' ingesteld op de naam 'main'. Van deze nieuwe tag <main> is het de bedoeling dat deze de typische 'hoofd' div gaat vervangen.<br>  Er mag maar één main element op een pagina zijn. Tevens mag deze geen onderdeel zijn van de hieronder volgende elementen. |
| Article | Dit is een element wat gebruikt wordt om aan te geven dat een stuk content bijvoorbeeld een blog (article) of forumpost is. Hierdoor zijn daaraan vanuit CSS, maar zeker ook vanuit JavaScript regels aan te koppelen die het werken met dit soort items duidelijker maakt. |
| Aside | Het aside element is bedoeld om een balk aan de zijkant van de webpagina mee te specificeren. Denk aan een blok met contactinformatie, of bijvoorbeeld een zogeheten wordcloud met daarin hot items waarop men kan klikken. |
| Section | De tag <section> geeft aan dat elementen die hierin gezet worden bij elkaar horen of gerelateerd zijn aan elkaar. Zoals bijvoorbeeld een kopje, een aantal paragrafen en wat plaatjes, die tezamen iets zeggen over het thema treinen. Zie het als een div, maar dan met deze beschreven betekenis.<br>  De <article> en <section> tags mogen meermaals binnen elkaar genest worden. |
| Header | De betekenis van het header element spreekt al redelijk voor zich, hoewel deze niet verward dient te worden met het head element met de <head> tag, waarvan wij effectief niets zien op de pagina. Nee, dit element met <header> tag kan meermaals binnen een webpagina worden gebruikt om inleidende informatie te geven over bijvoorbeeld een hele pagina, een sectie of een artikel. |
| Footer | Een footer element is de tegenhanger van een header element. Ook dit element mag vaker voorkomen binnen één webpagina. Dit element is bedoeld voor informatie over bijvoorbeeld: schrijver van een artikel of webpagina, de algemene voorwaarden, copyright informatie, enzovoorts. |
| Nav | Als laatste, maar niet minst belangrijke: het nav element met gelijknamige tag. Hierin kunnen wij 'blokken' met navigatie links kwijt. Daarbij geldt dat het element bedoeld is om navigatie-delen van een webpagina te beschrijven. Dus niet iedere link op een pagina hoort in een <nav> tag thuis. |

Onderstaand vind u een schematisch voorbeeld van welke elementen wél, en welke elementen niet binnen elkaar gebruikt mogen worden:




In het volgende voorbeeld zijn de hierboven getoonde elementen toegepast in de vorm van een 'review website'. Articles hebben wij daarin weergegeven met een zwart kader, sections met een rood kader en paragrafen met een blauw kader er omheen.

U bevindt zich nu hier: home > productreviews

## Review Festool Invalzaag

Gepost door:**Marco**
Gepost op:**10 oktober 2016**

### Unboxing

Het uitpakken was een enerverend proces, waarbij mijn ogen rijkelijk werden getrakteerd op een royale hoeveelheid knisperend piepschuim.

### Aansluiten

Toen eenmaal de machine was uitgepakt, was het aansluiten ervan een fluitje van een cent.

### Werken

De soepelheid waarmee de as het zaagblad aandreef was bijna beangstigend. Als ware het een zeeleeuw op het ijs, die gratieus een einde wilt gaan brengen aan de levenscyclus van een dartelende pinguin.

**Gerelateerde links:**
Verkoop site
Copyright informatie

De bijbehorende htmlcode:

```
<body>
  <nav>
    U bevindt zich nu hier:  home > productreviews
  </nav>
  <main>
    <h1>Review Festool Invalzaag</h1>
    <article>
      <p>
        Gepost door:<strong>Marco</strong><br/>
        Gepost op:<strong>10 oktober 2016</strong>
      </p>
      <section>
        <p>
        <h2>Unboxing</h2>
        Het uitpakken was een enerverend proces, waarbij mijn ogen rijkelijk werden
getrakteerd op
        een royale hoeveelheid knisperend piepschuim.
        </p>
        <p>
        <h2>Aansluiten</h2>
        Toen eenmaal de machine was uitgepakt, was het aansluiten ervan een fluitje
van een cent.
        </p>
        <p>
        <h2>Werken</h2>
        De soepelheid waarmee de as het zaagblad aandreef was bijna beangstigend.
        Als ware het een zeeleeuw op het ijs, die gratieus een einde wilt gaan
brengen
        aan de levenscyclus van een dartelende pinguin.
        </p>
      </section>
    </article>

    <aside>
      <span>
        <strong>Gerelateerde links:</strong><br/>
        Verkoop site
      </span>
    </aside>

  </main>
  <footer>
    Copyright informatie
  </footer>

</body>
```

Schrijf een eenvoudige HTML webpagina over een willekeurig thema dat u aanspreekt. U mag hiervoor kiezen om:
- een fysiek .html bestand te maken (en deze zelf te openen in de browser)
- een CodePen.IO 'Pen' te maken

Gebruik op de pagina minimaal: een article, een header, een footer en een nav element. Geef de pagina ook een toepasselijke title.

Test uiteraard of je pagina werkt.
Deze oefening is extra, mocht je er niet uit komen, neem dan contact op met de docent.

# 2 Elementen en attributen

## 2.5 Attributen

We hebben nu gezien dat elementen een webpagina opbouwen. Door het geven van een waarde aan bepaalde eigenschappen van een element, kunnen de karakteristieken van dat element nog nauwkeuriger worden bepaald. Dergelijke eigenschappen worden attributen genoemd. Een attribuut wordt alleen binnen de begin tag van een element opgenomen. Bij meerdere attributen is de volgorde overigens niet van belang.

We kunnen bijvoorbeeld een invoerveld plaatsen op een pagina met behulp van <input />. By default verschijnt er dan een invoerveld. De meest correcte notatie om een invoerveld te maken is echter: <input type="text" />. Waarom? Er is de mogelijkheid gegeven om van het elementtype input meerdere 'smaken' te maken. Dus wanneer wij het attribuut 'type' vullen met de waarde 'checkbox' in plaats van de waarde 'text', zullen wij geen invoerveld te zien krijgen, maar een veld waarin wij een vinkje kunnen zetten. Zo zijn er nog véél meer mogelijkheden m.b.t. invoervelden, maar die zullen in hoofdstuk 6 aan bod komen.

Sommige elementen hebben helemaal geen attributen. Voorbeelden zijn de fysieke tekstopmaak-elementen b, i en sub. Daarnaast: wanneer wij bijvoorbeeld een stuk JavaScript code aan onze pagina willen toevoegen, moeten wij het <script> element gebruiken. Voorheen diende daar nog een attribuut genaamd 'language' aan toegevoegd worden met als waarde 'javascript'. Dat hoeft inmiddels niet meer.
Soms komt het voor dat een attribuut geen waarde heeft. Het opnemen van het attribuut is dan al voldoende. Bijvoorbeeld:
<input disabled>
Dit zorgt ervoor dat dit input element momenteel niet bruikbaar is. Een andere (W3C) geldige syntax hiervoor zou zijn:
<input disabled="disabled">

Hoe zien verschillende <input> typen eruit?

<input type="text" /> met als resultaat:

<input type="text" disabled /> met als resultaat:

<input type="checkbox" /> met als resultaat:

Er zijn een aantal afspraken wat betreft de notatie van elementen (tags) en attributen:

- Tags worden in kleine letters opgenomen (niet verplicht).
- Attributen worden in kleine letters opgenomen (niet verplicht).
- Waarden van attributen worden (in kleine letters) tussen dubbele óf enkele quotes opgenomen.
- Attribuut en attribuutwaarde worden gescheiden door een is-gelijk-teken (=).

### 2.5.1 Generieke attributen

Binnen HTML 5 zijn er een aantal attributen die op alle elementen toepasbaar zijn. Onderstaand volgt een opsomming van deze elementen:

- Accesskey

Wordt gebruikt om middels een toetsencombinatie naar een bepaald element (vaak een invoerveld) te kunnen verwijzen.

Gebruik in het volgende voorbeeld de toetsen combinatie <alt><x> om naar het input veld te navigeren.

```
<html>
<head><title>Voorbeeld</title></head>
<body>
   <input type="text" accesskey="x" />
</body>
</html>
```

- **class**

Eén van de meest gebruikte attributen op HTML elementen. En eigenlijk wordt dit attribuut nog het meest gebruikt door Javascript en CSS code. Het attribuut is bedoeld om verschillende elementen te categoriseren.

- **contenteditable**

Met dit attribuut kunnen wij aangeven of het huidige element waar dit attribuut bij hoort, gewijzigd mag worden. Dat is dus erg leuk om dynamisch te wijzigen als reactie op een actie van een gebruiker. Dat kunnen wij doen middels Javascript code, want in onze Javascript cursus behandeld wordt. Dit element is binnen een element òf afwezig òf aanwezig (eventueel met als waarde 'contenteditable'.

- **contextmenu**

Een verwijzing naar een <menu> element met <menuitems> daarin, die gebruikt worden als 'rechtermuisknop' menu wanneer iemand met een rechtermuisknop op het element klikt. Vooralsnog wordt dit alléén ondersteund in Mozilla Firefox.

- **data-***

Deze groep attributen wordt veel gebruikt als manier om applicatie-specifieke HTML attributen te maken. Dit wordt onder meer toegepast om privédata op te slaan binnen de pagina of applicatie. Wij kunnen dus zelf een attribuutnaam bedenken en gebruiken die begint met 'data-'. Hierdoor kunnen wij bijvoorbeeld met CSS of JavaScript code heel eenvoudig naar specifieke elementen verwijzen met dit specifieke attribuut. Voorheen werden hier ook 'hidden fields' voor gebruikt (invoervelden met een waarde 'hidden' voor het 'type' attribuut. Echter was dat niet heel veilig wanneer data werd verzonden via een zogeheten 'GET' methode (hierover in een later hoofdstuk meer).

```
<html>
<head><title>Voorbeeld</title></head>
<body>
  <h2>Vul hier informatie in over uw Raspberry Pi:</h2>
  <input type="text" id="ipadres" data-categorie="raspberry" />
  <input type="button" id="verzendbutton" value="Toon data
attribuut waarde" />
<script>

      document.getElementById("verzendbutton").addEventListener(
                  "click", function(){
                              current =
document.querySelectorAll("input[data-categorie]")[0];
                              attribuut =
current.getAttribute("data-categorie")
                              alert("Waarde: " + attribuut);
                        });
</script>
</body>
</html>
```

- **draggable**

Geeft aan of een element versleepbaar is. Dit dient aan te staan om 'drag & drop' functionaliteit mogelijk te maken. Hierover lees je in een later hoofdstuk meer.
hidden
Dit attribuut kunnen wij instellen op 'yes' of 'no'. Daarmee geven wij aan of het element momenteel wel of niet aanwezig dient te zijn. Dit kan erg handig zijn in een formulier, bijvoorbeeld dat een invoerveld pas zichtbaar wordt zodra een gebruiker op een knop heeft geklikt. De instelling kan dan achter de schermen middels JavaScript van 'no' naar 'yes' worden gezet.

- **id**

Variant op het veel gebruikte class attribuut. Wij kunnen aan de hand hiervan een unieke naam of referentie geven aan een element. Deze verwijzing is vervolgens binnen JavaScript en CSS weer te gebruiken. Let op: er mag binnen een enkele webpagina maar één element zijn met een specifiek id. Dit is daarin tegen bij een class attribuut niet het geval.

- **style**

Willen wij met CSS hier een stijl aan toevoegen? Zo ja, dan zetten wij deze opmaak rechtstreeks in het style element.

- **tabindex**

Hiermee geven wij een nummer op in de zogeheten 'tab volgorde' binnen een pagina. Wanneer wij op een webpagina op <TAB> drukken, zal onze cursor een element verder springen. Naar welk volgend element zal hij moeten springen? Logischerwijs bijvoorbeeld na een postcode, willen wij vaak dat de cursor na het drukken op <TAB> dan door gaat naar het huisnummer en niet naar achternaam. Bekijk hiervoor het volgende voorbeeld:
Het onderstaand toegepaste element form gebruiken wij om aan te geven dat alle elementen die daarin staan, binnen een gezamenlijk formulier vallen. Meer hierover in het hoofdstuk 'formulieren'.

Let op: Wij hebben bewust de elementen in een onlogische volgorde gepositioneerd. Het doel van de tab-index is om er in ons voorbeeld in de volgende volgorde doorheen te kunnen navigeren middels <tab>: naam, postcode, huisnummer.

```
<html>
<head><title>Voorbeeld</title></head>
<body>
  <form>
            Huisnummer: <input type="text" id="huisnummer"
tabindex="3" /><br/>
            Postcode: <input type="text" id="postcode"
tabindex="2" /><br/>
            Naam: <input type="text" id="naam" tabindex="1"
/><br/>
            <input type="button" id="verzendbutton"
tabindex="4" value="Knop" /><br/>
      </form>
      <script>
      //bij het laden  krijgt het element naam  de focus
      document.getElementById("naam").focus();
      </script>
</body>
</html>
```

- **title**

Geeft extra informatie over een element, wat vaak door zoekmachines gebruikt wordt als context.

- **translate**

Willen wij de inhoud van dit element eventueel kunnen (laten) vertalen?

Een aantal andere mogelijke attributen die minder gebruikt worden zijn: dir, lang, spellcheck en dropzone (wordt niet ondersteund).

Tot slot
Het toepassen van attributen zal in de cursus veelvuldig aan bod komen. In de cursus kunnen we nooit alle bestaande elementen en attributen behandelen. De meest gebruikte elementen zullen gaan wij in ieder geval bekijken. Daarnaast zullen we niet altijd alle attributen van een element opsommen, maar wél degenen die in de praktijk het meest toepasbaar zijn.

### 2.6 Samenvatting

Wij hebben in dit hoofdstuk geleerd wat elementen zijn en welke elementen de essentie van een webpagina vormen. Wij hebben gezien welke elementen zorgen voor een semantisch goed opgebouwde pagina, met HTML 4, maar ook met HTML 5. Daarnaast hebben wij gekeken welk effect blok elementen hebben ten opzichte van inline elementen en welke typen wij wel of niet binnen elkaar kunnen nesten. Tot slot hebben wij geleerd wat attributen zijn, wat de meest voorkomende attributen zijn en hoe die toe te passen op het merendeel van elementen.

# 3 Opmaak

## 3.1 Inleiding

Opmaak werd voorheen altijd gedaan met behulp van opmaak elementen. Deze gaven dan een gekozen lettertype, kleur of andere eigenschap aan. Met de komst van HTML 4 en daarna 5 worden deze elementen sterk afgeraden om te gebruiken. Er zijn nog wél semantische elementen die iets vertellen over wat voor een soort tekst wij mee te maken hebben. Dáár gaan wij zo dieper op in.

Ook zullen wij kijken naar hoe wij CSS code kunnen toevoegen aan onze HTML file. Het opmaken van een webpagina gebeurt immers voor het grootste deel met CSS. Maar zoals gezegd gaan wij in deze cursus niet te ver in op CSS.

- Fysieke en logische (semantische) elementen kennen voor het opmaken van pagina's met HTML
- Ervoor kunnen zorgen dat een webpagina CSS code gaat gebruiken3

## 3.2 Fysieke elementen

Sinds enige HTML versies is het mogelijk om middels een aantal elementen aan te geven of je bijvoorbeeld een stuk tekst dikgedrukt, schuin gedrukt of onderstreept wilt weergeven. Deze elementen zijn nog mogelijk om te gebruiken, maar worden niet meer aanbevolen door het W3C. Welke elementen zijn dit dan?

- <b> wordt gebruikt om aan te geven dat de tussenliggende tekst dik gedrukt moet worden weergegeven indien het gebruikte lettertype dit ondersteund.
- <i> kunnen wij gebruiken om een tekst schuin gedrukt, ook wel italic, weer te geven. Dit is afhankelijk van de mogelijkheden van het gebruikte lettertype.
- <u> ook wel underline kan de tekst, mits ondersteund door het lettertype, onderstreept weergeven. Underline wordt overigens niet aangeraden om veel te gebruiken. Woorden of teksten die onderstreept zijn kunnen snel verward worden met hyperlinks (snelkoppelingen).
- <br /> dit element is te gebruiken om een regelovergang aan te geven. Wordt erg veel gebruikt, omdat we een zelf getypte 'enter' niet terug zien in de output.

De elementen <b>, <i> en <u> worden na de tekst ook weer afgesloten met de bijbehorende sluit-tags.

```
<b>dik gedrukt</b>          dik gedrukt
<i>schuin gedrukt</i>       schuin gedrukt
<u>onderstreept</u>         onderstreept
<u>lijkt op: <a href="#">hyperlink</a>    hyperlink
```

Het voorbeeld illustreert dat we dus met HTML heel directe tekst opmaak commando's (tags) kunnen gebruiken. Dit strookt echter niet met het idee van HTML 5, dat HTML elementen zoveel mogelijk semantisch zijn. Waarschijnlijk heeft dit te maken met backwards compatibility.

# 3 Opmaak

## 3.3 Semantische opmaak elementen

Naast de reeds behandelde semantische elementen, is er ook een groot aantal elementen die iets zeggen over de betekenis van een tekst. Daar kunnen vervolgens door de browser (default instellingen) óf door CSS code stijlen aan gekoppeld worden.

We beginnen met: <small>, wat een tekst minder belangrijk maakt, evenals <sub>, wat een subtekst impliceert. Vaak hebben beide als gevolg dat een tekst (by default) kleiner weergegeven wordt dan het standaard lettertype. Daarnaast wordt een tekst die omringd is met <sub> tags vaak een halve regel lager getoond dan de omringende tekst. Als tegenhanger is er <sup>, wat staat voor supertekst. Deze tekst wordt weer vaak een halve regel hoger getoond. Beiden kunnen handig zijn om chemische of wiskundige notaties weer te geven zoals:

Een teken dat gebruikt kan wordem om uranium weer te geven, is: $U^{92}$.
De weergave van water als chemische afkorting is: $H_2O$.
De oppervlakte van een huis van 11 meter lang, bij 5 meter breed is: $55m^3$
Dit is een tekst om te demonstreren hoe de b van **bold**, de i van *italic* en de u van <u>underline</u> eruit zien.

Probeer om zelf output  met opmaak in HTML code te maken zoals getoond op de vorige pagina. Kijk ook eens of het lukt om een woord of zin zowel *italic* als **bold** te krijgen. Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

Er zijn een aantal elementen om aan te geven dat een tekst benadrukt wordt. Vaak wordt daarvoor het i element gebruikt. Maar we kunnen een aantal andere elementen onderscheiden, die wellicht nog duidelijker zijn in de semantiek die ze bieden:

-  Is een afkorting die staat voor *emphasis*, nadruk dus.
-  kunnen we gebruiken om iets als belangrijk te markeren.
- <mark> Is een leuke toevoeging waarmee wij kunnen aangeven dat iets als het ware gemarkeerd moet worden.
- <q> wordt gebruikt om tekst te quoten.
- <cite> wordt net als <q> gebruikt om een citaat mee te beschrijven
- <blockquote> Is innig verweven met het <cite> element. Vaak wordt een <cite> element binnen dit element toegepast. Als volgt:

```
<blockquote>
Met dank aan mijn vader voor alle hulp en wijsheid die hij
bied en heeft geboden.<br/>
En zoals hij altijd zegt in voor en tegenspoed: <br/>
    <cite>
      C'est la vie!<br/>
     </cite>
</blockquote>
```

Uitkomst:
Met dank aan mijn vader voor alle hulp en wijsheid die hij bied en heeft geboden.
En zoals hij altijd zegt in voor en tegenspoed:
*C'est la vie!*

- <h1> t/m <h6> gebruiken wij om kopjes mee te definiëren. In deze orde van groot naar klein. Erg handig om onderscheid te maken tussen titels, wanneer wij enkel <b> zouden hebben gebruikt bijvoorbeeld.
- <code> Wordt veel op fora en websites zoals stackexchange gebruikt, om te tonen dat iets om een code voorbeeld gaat. Erg prettig om hiermee de (bijvoorbeeld) programmacode te onderscheiden van normale tekst.
- <ins> en <del> kunnen respectievelijk gebruikt worden om aan te geven dat deze tekst is toegevoegd of verwijderd uit een tekst zoals een artikel.
- <details> Zorgt voor een uitklapbaar stukje tekst, waardoor de gegevens die je binnen dit element zet, standaard niet zichtbaar zijn op het scherm, totdat de gebruiker op het pijltje klikt. Zeer handig wanneer er bijvoorbeeld voor kinderen gevoelige informatie staat die niet direct op het scherm getoond mag worden bij het laden van de pagina, óf wanneer er op een puzzel website, de oplossing van een puzzel staat, terwijl iemand enkel wat tips in de goede richting wilt.

```
<body>
  <main>
    <article>
      <section>
        <h2>Review</h2>
        <p>Uitgegeven op de SNES spelcomputer, maar nu
jaren later nog door vele mensen gespeeld.</p>

        <h3>Tip's & tricks:</h3>
        <details>
          Vergeet niet dat u de spiegel kunt gebruiken om
terug te keren naar de 'light world'.
        </details>
      </section>
    </article>
  </main>
</body>
```

Resultaat; links ingeklapt en rechts uitgeklapt:


- <time> In een time element kan een tijdstip worden geplaatst. Deze tijd dient niet als tekst in het element zelf, maar in het bijbehorende (maar niet verplichte) attribuut 'datetime' geplaatst te worden. Ook dient dit tijdstip te zijn opgemaakt volgens een

vooraf gedefinieerd formaat. De tijd die is opgeslagen in het datetime attribuut, kan door de computer gebruikt worden om bijvoorbeeld de lokale tijd op te slaan voor gebruik in een evenement. Denk bijvoorbeeld aan een knop die een evenement toevoegt aan de Google Agenda van de eindgebruiker.

Het <time> element zelf mag ook tekst bevatten, maar daarmee zal de computer zelf niets doen. Die tekst is dan enkel bedoeld voor de bezoeker van een webpagina.

Op vrijwel alle hiervoor genoemde elementen wordt door browsers een default opmaakt toegepast. In de praktijk is het vaak de bedoeling deze elementen middels CSS een toepasselijke stijl te geven.

Tot slot: het is vaak handig om bij uw code commentaar te plaatsen om bepaalde stukken code te verduidelijken (als de code niet voor zich spreekt). Ook kan het handig zijn om even een regel 'uit' te kunnen zetten, waardoor die niet geïnterpreteerd wordt en dus niet uitgevoerd en/of getoond wordt.

Dit kunnen wij binnen HTML bewerkstelligen met de <!-- en --> tags. Commentaar typen we dan tussen deze twee tags in. Let vooral op het uitroepteken (!) bij de eerste tag.

```
<div>
    <!-- hier komt straks nog een paragraaf in -->
</div>
```

Probeer in één van de eerdere opdrachten zelf ook commentaar toe te voegen, óf, één gehele regel bestaande code in commentaar te zetten en kijk wat er gebeurd.
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

## 3.4 Webpagina's opmaken met Cascading Style Sheet (CSS)

De norm is tegenwoordig om webpagina's op het vlak van indeling, kleuren, teksten, media zoals foto's, animaties, geluid en video middels CSS code op te maken.

Met CSS ('Cascading Style Sheets') verwijzen wij naar één of meer elementen en passen daar opmaak regels op toe. De kracht van CSS is dat je met relatief weinig code, véél verschillende elementen van een pagina kunt stijlen. Deze code is daarnaast eenvoudig toepasbaar op meerdere webpagina's, wat dus ook weer code kan besparen.

Zoals u eerder heeft kunnen lezen bevat HTML nog wel enkele niet semantisch opmaak elementen, maar het wordt aangeraden om deze niet meer te gebruiken.

Hoe ziet CSS code er eigenlijk uit?

Stel dat wij de achtergrond kleur van een div ( ) en het gebruikte lettertype binnen een paragraaf (<p>) willen wijzigen, dan kan dat middels CSS als volgt:

# 3 Opmaak

```
div {
  background-color: "yellow";
}

p {
  font-family: "Verdana";
}
```

De gebruikte dubbele quotes zijn pas verplicht wanneer wij een lettertype of kleur benoemen die uit meer dan één woord bestaat. Het aanleren om de quotes erbij te typen kan dus geen kwaad.

Binnen CodePen kun je je CSS code kwijt in het 'CSS' veld. In een fysiek HTML bestand, zul je de code in een los .CSS bestand of bovenin je HTML bestand tussen <style> en </style> tags plaatsen.
CSS code kan heel modulair toegepast worden binnen één of meerdere webpagina's. Gangbaar is dat CSS code in een aparte .css file wordt opgeslagen. Vervolgens kan er vanuit één of meerdere webpagina's een verwijzing naartoe gemaakt worden. Ook is het mogelijk meerdere .css files aan een document te koppelen, zoals één die gebruikt wordt voor weergave op een computer, één voor weergave op een mobiele telefoon en één die de opmaak toonbaar maakt voor op papier (na afdrukken).

Onderstaand lezen we de vier manieren om CSS code op een webpagina toe te passen:

- **Embedded**, dat wil zeggen: in de HTML pagina verwerkt, maar wél buiten de beschrijving van de elementen. Vaak wordt bovenaan de pagina een HTML codeblok gemaakt, waarin wij onze CSS code kwijt kunnen.
  ```
  <style type="text/css">
  p { color: green; }
  </style>
  ```
- **Inline**, zoals de naam al zegt: in de regel. Dus via deze weg kun je CSS code als zogeheten style attribuut toekennen aan een willekeurig HTML element. Kort en snel, maar niet flexibel of beheerbaar.
  ```
  <p style="color: green">
  ```
- **Linked**, is de meest gebruikte manier voor het toekennen van CSS code aan een HTML pagina. Er wordt middels een <link> tag met een src attribuut, verwezen naar een .css bestand dat op de betreffende pagina moet worden toegepast.
  ```
  <link rel="stylesheet" type="text/css" href="naam.css"
  media="all">
  ```
- **Imported**, is een variant op de bovengenoemde linked toepassing. Het importeren van een stylesheet gebeurt hier met het @import commando, binnen <style> HTML element tags.
  ```
  <style type="text/css">
  @import url(bestandsnaam) mediatype;
  </style>
  ```

De linked en imported toepassingen zijn erg veelzijdigomdat wij een media type kunnen opgeven waarvoor deze specifieke .css file gebruikt zal gaan worden. Denk aan een speciale .css file voor het printen van een webpagina, die wij aanroepen met: media="print".

Het toepassen van CSS bestanden op HTML files op een mobiel device kan op verschillende manieren. Wilt u daar graag meer over weten, bekijk dan onze Cascading Style Sheets training. De volgende website is ook een handige opstap hiervoor: Return of the mobile stylesheet (http://alistapart.com/article/return-of-the-mobile-stylesheet)

### 3.5 Samenvatting

Wij hebben gezien welke elementen toe te passen zijn om de opmaak van tekst en de semantische opbouw van een webpagina tot op zekere hoogte aan te passen. Ook is in dit hoofdstuk is aan bod gekomen hoe wij CSS code kunnen toepassen binnen onze webpagina('s) en wat de voordelen daarvan zijn.

# 4 Afbeeldingen, hyperlinks en embedding

## 4.1 Inleiding

In dit hoofdstuk zien we hoe we plaatjes en hyperlinks kunnen toevoegen aan een webpagina. We gaan kijken naar het embed element en we bekijken we een andere mogelijkheid om een pagina te structureren, namelijk met frames.

- Het kennen van de verschillende tags om afbeeldingen mee weer te geven in HTML
- Het weten van de verschillen tussen het alt en title attribuut van een afbeelding
- Hyperlinks kunnen maken naar andere pagina's, maar ook naar ankers binnen dezelfde pagina
- Zelf de <embed> tag kunnen toepassen binnen een webpagina

## 4.2 Afbeeldingen met img element

Het opnemen van plaatjes kan een webpagina aantrekkelijk maken. Met behulp van de <img> tag kan een plaatje in de pagina worden opgenomen. Het img element is een inline element, dus er kan bijvoorbeeld gewoon tekst naast getypt worden. Maar in tegenstelling tot 'normale' inline elementen, mag een afbeelding wél een opgegeven breedte en hoogte hebben.

Qua bestandsformaten hangt de ondersteuning af van uw browser. Een greep uit de mogelijkheden: jpg, tiff, bmp, svg en png. Maar ook nieuwere formaten zoals jxr en webp.

Kort beschreven ziet de opbouw van een img element er als volgt uit:

<img src="galaga.jpg" title="Dit is een plaatje van het spel Galaga" />

Is het opgevallen dat het img element met een slash wordt afgesloten? Dat betekent dat er in een image geen andere element of tekst mag komen te staan. Feitelijk wordt eigenlijk een <img /> tag achter de schermen letterlijk vervangen door de bit representatie van een fysieke afbeelding.

Het img element heeft een aantal attributen, waarvan we de belangrijksten even opsommen:

| | |
|---|---|
| Src | De bestandsnaam (source) van de afbeelding of een zogeheten datastream. Ondersteunde formaten van afbeeldingen zijn: jpeg, gif en png. |
| Alt | De tekst die in plaats van de afbeelding geplaatst wordt, als de browser geen afbeeldingen kan weergeven. Deze tekst kan ook de vindbaarheid van een webpagina bevorderen en is de 'gesproken' tekst bij gebruik van een screen reader apparaat. |
| Height | De hoogte van de afbeelding (in pixels). |
| Width | De breedte van de afbeelding (in pixels). |
| Title | De inhoud van het title attribuut is de waarde die de gebruiker ziet wanneer hij of zij met zijn of haar muis op de afbeelding blijft staan. |

| | |
|---|---|
| Crossorigin | Wordt gebruikt om een plaatje vanaf een andere bron (server; website) te laden in een <img> element. Daarvoor is tevens Javascript code nodig. |
| Ismap | Kan als attribuut aanwezig of afwezig zijn. Het attribuut heeft zelf geen waarde. Dit plaatje wordt hierdoor gezien als een zogeheten 'image-map'. Dat zorgt er in dit geval voor dat na klikken op het plaatje, de exacte coördinaten van de klik positie naar de server gestuurd worden voor verdere verwerking. |
| Usemap | Hier geven wij een naam van een bestaande imagemap op waarnaar wij verwijzen. Een imagemap bevat alleen regio's en coördinaten, geen plaatje. Hiermee wordt dus dit plaatje aan een map gekoppeld. |

Het opgeven van de afmetingen met width en height bespaart laadtijd, omdat de browser het voor de afbeelding benodigde gebied vooraf kan berekenen en met het plaatsen van de rest van de inhoud van het document niet hoeft te wachten tot de afbeelding geheel geladen is. Wanneer we een plaatje proportioneel willen schalen, kunnen we alleen height of alleen width opgeven. Het andere attribuut wordt dan in dezelfde verhouding vergroot of verkleind.

Van bovengenoemde attributen is het src attribuut de meest belangrijke. Hiermee geven wij het (relatieve) pad óf de website URL aan van het plaatje dat wij willen zien.

*Absoluut pad.* De directory die is opgegeven vanaf de zogeheten root van een schijf. In Windows-kringen wilt dat zeggen: het gehele pad inclusief schijfletter. In Linux kringen wilt dat zeggen: het gehele pad, gezien vanaf de root.

Bijvoorbeeld:
```
c:\games\zelda\tingle.jpg (onveilig, niet te gebruiken in src attribuut)
https://upload.wikimedia.org/wikipedia/en/d/d7/Tingle.png  (prima te gebruiken)
```

*Relatief pad*. De directory die is opgegeven bekeken vanuit ons huidige locatie. Hierbij mag geen drive-letter worden opgegeven en is dus nooit het hele pad naar de file toe, slechts vanuit onze huidige directory het pad naar de file toe.

Bijvoorbeeld:
```
images/games/zelda/tingle.jpg
```

Het opgeven van het pad binnen het src attribuut kan op de volgende verschillende manieren:

voorbeeld:

Relatief pad opgeven. Er wordt gezocht naar de file 'fictief' in de directory 'images', bekeken vanuit de huidige directory van de webpagina:
```
<img src="images/fictief.jpg" />
```
Met een absoluut pad opgegeven, kan een plaatje gebruikt worden dat niet op deze huidige server staat. Als dat wél had gekund, dan was het voor hackers heel eenvoudig geweest om achter de directory structuur van een server te komen:
```
<img src="http://www.vijfhart.nl/fictief.jpg" />
```
Let op: Het volgende mag dus niet (zou niet veilig zijn en kán daarom ook niet):
```
<img src="c:\data\private\web\fictief.jpg" />
```

# 4 Afbeeldingen, hyperlinks en embedding

Indien we naar een plaatje of webpagina willen verwijzen middels een URL, maar we willen zeker weten dat er hetzelfde protocol gebruikt wordt, dan kunnen we het absolute pad beginnen met: //, dus zonder de http óf https ervoor. Wanneer kan dat handig zijn? Indien je de website gebruikt via het https protocol, dan wil je liever geen verwijzingen naar pagina's of bestanden via het http protocol. Bij gebruik van enkel de dubbele slash (//) wordt gekeken naar wat het *huidige* protocol is en die wordt toegepast op de URL.

### 4.2.1 datastreams

Wat tevens laadtijd kan besparen is het gebruik van een datastream in het 'src' attribuut. We vullen dan dus geen absoluut of relatief pad in, maar een een verwijzing naar een plaatje middels een zogeheten datastream.

Met behulp van tools is een plaatje om te zetten naar een datastream. De image file wordt hierdoor rechtstreeks in de webpagina geladen. Dit kan een voordeel of een nadeel bieden, afhankelijk van de omvang van de file. Doordat het bestand niet meer door de site bezoeker gedownload (en e.v.t gecached) hoeft te worden, scheelt dit een zogeheten 'http-request'. Maar, bij gebruik van een groot plaatje, wordt de bestandsgrootte van de .html file weer erg groot, wat het laden van de algehele webpagina kan vertragen. Uit ervaring moet blijken wat interessanter is.
Onderstaand volgt een voorbeeld van het toevoegen van een plaatje aan uw website, met behulp van een datastream:

```
<html>
<head><title>Voorbeeld</title></head>
<body>
            <img src="
AAD/2wBDAAoHBwgHBgoICAgLCgoLDhgQDg0NDh0VFhEYIx8lJCIfIiEmKzcvJik0KSEiM
EExNDk7Pj4+JS5ESUM8SDc9Pjv/2wBDAQoLCw4NDhwQEBw7KCIoOzs7Ozs7Ozs7Oz
s7Ozs7Ozs7Ozs7Ozs7Ozs7Ozs7Ozs7Ozs7Ozs7Ozv/wAARCAAgACADASIAAhE
BAxEB/8QAFwAAAwEAAAAAAAAAAAAAAAABAUGB//EACgQAAEEAQMEAQUBAQAAAAAAAAEC
AwQRBQYSIQATMUFRByJCgZEjM//EABkBAAIDAQAAAAAAAAAAAAAAAAMFAQIEBv/EACgRA
AIABAQEBwAAAAAAAAAAAAECAAMEEQUSITETQVFhFCIygaHw8f/aAAwDAQACEQMRAD8AY6
61pPdzDmNxkpyLHhrKVuMrUhbixwqzwaBsV4NXzxRcSJrvTmWTKkqZSE2pKXgh8vhaCR
e1BIVuHzQ8fF3KY2A/i9cQYMlO11jINIVwQD94oi64I5B+COtP1vqh7TMCMuIllcl90gI
eQogoA+48EcglPv30sQ580xyRYx3FUop+DR00tXDqdxv3vuOZvy06QNrTWM3Tym2oOPU4
dw7kiQ0vsgkWEpIrcr354r2boHTH1JTkpiIOWjpZffdCGXI6TsJPACgSSDdCxfnmqvoSN
9W/wDkmVh/gOuNP/0pSU/wE/v31GalyMTLaglz4LKmWH1JUEKSEm9o3EgEiybP76mZUWO
dGv2itHhAeWaeokZTb1g31v8AdOkarqqImBJjarjxVSZONSUuNBZTvaIUD6PKSrdfxuv1
Spev9IZd9pvI49xSU3tclxUOJb458FR5oeB8dXDjbbzS2nUJcbWkpUhQsKB8gj2OsfzOg
su3qGTFxePcein/AFZXYSgIP47lHyCaomzV10afxEN0FwdxaFmFeFqV4dSxVkHlbNbTp7
E6djFbnfp7hsljnJmGb7MlSFOs9hy2nieQKJoA+BRAF+wK6W6F0LKjzm8vl2uz2qVHjrA
KiSOFKH41fA82PVc3GBxicPg4ePASFMNAL2kkFZ5UQTzRUSf30w6IKdCwcixjK2L1KSnp
lfMpJAJ3tt8/kf/Z" alt="5hart-it logo" />
</body>
</html>
```

Resultaat:

Het encoderen van een afbeelding zoals bovenstaand gebeurt niet vaak. Voor zware professionele websites, die moeten omgaan met vele duizenden bezoekers en pagina-laad acties, kan dit interessant voor zijn. Daarom zullen wij dit verder ook niet behandelen in deze cursus. Mocht uw interesse hierin toch zijn gewekt en u wilt zelf afbeeldingen omzetten naar het zogeheten base64 datastream formaat, dan is dat niet iets wat wij eenvoudig met de huidige basiskennis kunnen doen. Een website die deze klus bijvoorbeeld voor ons kan klaren is: Base64online. (http://base64online.org/)

We gaan nu zelf oefenen met het gebruik van het img element met een src attribuut.

Probeer onder staande code uit en probeer te achterhalen hoe elk element door de browser wordt geinterpreteerd;

```
<html>
<head><title>Voorbeeld</title>
 <style>
 body {
  background-color: red;
}
 </style>
</head>
<body>
    <h1>Het ophalen van een plaatje : </h1> <br/>
    <img src="https://www.vijfhart.nl/wp-
content/themes/vijfhart/dist/images/5hart-logo-white.png"
alt="5hart-it website logo" />
    Deze tekst komt naast het plaatje te staan omdat deze
na het plaatje in de code staat én omdat <img> een inline
element is.
</body>
</html>
```

Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

Middels het alt attribuut stellen wij een tekst in die verschijnt wanneer de browser de afbeelding niet kan laden. Het wordt aangeraden om altijd een alt-waarde mee te geven aan een afbeelding. Bij een trage internetverbinding kunnen we de browser zo instellen dat afbeeldingen niet automatisch geladen worden. De opgegeven alt-waarde wordt dan ook als tekst zichtbaar op de plek van het plaatje.

Daarnaast wordt het alt attribuut gelezen door o.a. de zoekmachines van Google om de indexeerbaarheid te vergroten. Dit is een niet te onderschatten reden om een goede alt tekst te kiezen die het plaatje verklaard en/of beschrijft. Gebruik dus goede keywords, die je ook gebruikt op de rest van je pagina.

Een derde toepassing ervan is dat dit attribuut gebruikt kan worden door screen-readers dat zijn stukken software om teksten op te lezen voor bijvoorbeeld mensen met een verminderd zichtveld.

Bij het volgende plaatje hebben wij een title attribuut ingesteld, om een waarde te tonen wanneer een gebruiker de muis boven op de afbeelding houdt. Ook hebben wij een beschrijvende alt-waarde opgegeven om hem beter vindbaar te maken voor Google.

```
<img
src="http://www.pokemongo.com/static/assets/images/pokemon_go_logo.png"
alt="Pokémon Go logo" title="De Pokémon rage is bijna voorbij"
width="200">
```

### 4.3 Gebruik van het figure element

anaf HTML 5 wordt ook gebruik gemaakt van de semantische elementen figure en het bijbehorende figcaption. Deze worden gebruikt om een figuur te identificeren, hier mag weer een voor zichzelf sprekend figuur komen te staan. Dus denk aan een foto, diagram of tekening. Het is netjes om in een figure eveneens een figcaption te gebruiken.

Qua structuur komt dat er grofweg uit te zien als:

```
figure>
    <img src="bronfile.gif" alt="alternatieve tekst"/>
    <figcaption>beschrijving</figcaption>
</figure>
```

```
<body>
      <figure>
            <figcaption>Metabo boorhamer</figcaption>
            <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/metabo.jpeg" />
      </figure>
      <figure>
            <figcaption>deWalt boorhamer</figcaption>
            <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/dewalt.jpeg" />
      </figure>
</body>
```

Probeer dit nu zelf. Maak een HTML file zoals de voorgaande en verwerk minimaal de elementen die nodig zijn voor een goede W3C validatie (https://validator.w3.org/), klik op 'validate by file upload' en blader naar uw webpagina. Voeg ook de elementen toe die nodig zijn voor het semantisch correct weergeven van een plaatje.

Wat vullen we in als waarde voor het attribuut src bij ons plaatje?

Ga daarvoor naar Google, typ daar een woord naar keuze in. Klik op 'Google Zoeken'. Als u resultaten ziet, klik dan op 'Afbeeldingen' om enkel op afbeeldingen te zoeken. Kies een afbeelding die u aanspreekt en klik er met uw rechter muisknop op en kies vervolgens 'adres van afbeelding kopiëren'. Je krijgt dan het pad van de afbeelding die je kunt plakken in het src attribuut van onze <img> tag.

Bedenk of je nog meer nuttige attributen kunt toevoegen aan het img element. Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

### 4.3.1 Opkomst van het picture element

Het picture element is tot op heden nog geen officieel onderdeel van de W3C HTML 5 specificatie. Echter,  dit element krijgt veel support van de community vanwege zijn meer dan praktische en eenvoudige toepasbaarheid.

Het *picture* element heeft gelijknamige begin en eindtags: <picture> en </picture>. Daarbinnen hoort minimaal één <source /> tag of <img /> tag gebruikt te worden.

Een *source* element kan binnen een *picture* element, maar ook binnen een *video* of *audio* element worden gebruikt. Over de laatst genoemden is meer te ervaren in hoofdstuk 7.

<Source> tag elementen worden gebruikt om weergave alternatieven op te geven bij onder andere verschillende device resoluties (mobiel, tablet, computer) en landscape variaties (telefoon rechtop of gekanteld). Ieder <source> element krijgt een eigen 'srcset' attribuut, waarin wij afbeelding opties aangeven voor apparaten met lage of hoge resoluties (https://developers.google.com/web/fundamentals/design-and-ux/responsive/images#enhance-imgs-with-srcset-for-high-dpi-devices). Ook is het gebruik om ieder <source> element een eigen 'media' attribuut te geven, waarin een zogeheten 'media query' wordt geschreven. We geven daarmee aan bij welke breedte van de schermweergave (denk aan handmatig verkleinen van venster door gebruiker) de browser dit source element dient te gebruiken.

Het is raadzaam om als laatste (onderste) optie binnen een <picture> tag, een terugval mogelijkheid te definiëren in de vorm van een img element voor het geval de browser niet om weet te gaan met een <picture> tag.

Onderstaand volgt een voorbeeld. Kijk of u in de Developer Tools onder 'network' kunt terug vinden welke file er geladen wordt bij het vergroten of verkleinen van het browser venster.

```
<body>
            <picture>
                  <source media="(min-width: 800px)"
srcset="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/5H_logo_groot_lowres.jpg
1x, https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/5H_logo_groot_hires.jpg
2x" />
                  <source media="(min-width: 400px)"
srcset="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/5H_logo_klein_lowres.jpg
1x, https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/5H_logo_klein_hires.jpg
2x"  />

                  <img src="5H_logo.jpg" alt="Vijfhart
Logo" />
            </picture>
```

```
</body>
```

Bekijk, indien u hier meer over wilt weten, ook eens de volgende cursus van Google, waarbij ook veel gebruik wordt gemaakt van CSS:
https://developers.google.com/web/fundamentals/media/images/

### 4.3.2 Responsive design en performance

Zowel responsive design als performance zijn noodzakelijk in een tijd als deze, waar webpagina's en applicaties perfect moeten kunnen draaien op devices van allerlei formaten (responsive). Momenteel wordt vrijwel altijd een plaatje geladen en vervolgens verkleind naar het formaat van het client device met behulp van CSS of Javascript code. Dit zorgde ervoor dat de bronfile (hoe groot deze ook mocht zijn) in zijn geheel gedownload moest worden bij de bezoeker en vervolgens verkleind werd. Tot voor kort was deze manier van werken noodzakelijk.

Gelukkig doet ook hier het element <picture> zijn intrede. Wij kunnen door gebruik te maken van het eerder genoemde mechanisme, in combinatie met de *media* (CSS media query) en *srcset* attributen, deze performance bottleneck tackelen.

Indien je van bovenstaande voorbeeld de performance hebt bekeken in je Chrome browser (via <ctrl><shift><i> - 'Network'), is het je wellicht opgevallen dat het plaatje dat beschreven staat bij de <img> tag, geladen wordt.

Waarom? Dit komt doordat het *picture* element nog niet volledig ondersteund wordt in je browser. De browser doorloopt dan een webpagina en zodra hij o.a. een <img> tag tegenkomt, zal hij het bijbehorende plaatje downloaden, ongeacht (voor alsnog) of dit element in een <picture> tag is opgenomen. In o.a. de moderne Chrome browsers (v.a. versie 53.027) werkt het echter zoals het hoort.

In HTML 5 kunnen wij met imagemaps een plaatje opdelen in sectoren en de browser anders laten reageren op het klikken op verschillende delen van het plaatje. Zie het als een computerspel waarbij je op zoek moet gaan naar 'geheime' objecten in een kamer, die je enkel kunt vinden door erop te klikken.

In de praktijk moet er een server zijn ingericht om een imagemap te kunnen testen. Wij kunnen wel de toelichting geven die nodig is om er zelf mee aan de slag te kunnen:

Er zijn een aantal zaken noodzakelijk om een imagemap te kunnen maken. Wij hebben nodig:

- Een plaatje met verwijzing naar een map element
- Een map element met daarin een set aan gedefinieerde vormen met bijbehorende coördinaten, die aangeven waar deze vormen zich bevinden ten opzichte van links bovenin het plaatje (0,0). De volgende vormen kunnen we identificeren in een map:
  - rect: een rechthoek, waarbij de x en y coördinaten van zowel de linker bovenhoek als de rechter hoek onderin opgegeven dienen te worden.
  - circle: een cirkel, waarbij de x en y coördinaten van het middelpunt van de cirkel en de lengte van de straal dienen op te geven
  - poly: een polygoon, dus een vorm die geheel zelf met coördinaten te formuleren is. Er wordt daarbij minimaal één x en y coördinaat verwacht. Bij meer coördinaten, worden deze punten onderling verbonden met een lijn om zo een polygoon te vormen.

# 4 Afbeeldingen, hyperlinks en embedding

```
<html>
<head><title>Voorbeeld</title></head>
<body>
            <map name="sectoren">
                  <area shape=rect coords="0,0 144,73"
onclick="alert('blauw');" alt="blauw" />
                  <area shape=rect coords="145,74 306,142"
onclick="alert('rood');" alt="rood" />
                  <area shape=rect coords="145,0 306,73"
onclick="alert('grijs');" alt="grijs" />
            </map>

            <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/map_plaatje.jpg" ismap
alt="image map voorbeeld" usemap="#sectoren"/>
</body>

</html>
```

Probeer nu zelf een HTML 5 bestand met daarin een imagemap te maken. Kies eerst een plaatje die op uw computer staat, of download een plaatje van Internet.

Tip: Maak eerst het bijbehorende img element op je webpagina en test of hij het plaatje laat zien. Werkt dat? Begin dán aan het element met de <map> tag.

Maak eerst alleen gebruik van een rechthoek als map deel in je plaatje. Het is daarnaast niet erg als u willekeurige coördinaten bedenkt voor de rechthoek.

Let op: Het is in de praktijk vrij arbeidsintensief om de juiste coördinaten op te zoeken. Gebruik hiervoor een tool zoals Adobe Photoshop, of vrij verkrijgbare software zoals Gimp. Zelfs Microsoft Paint is hiervoor te gebruiken.
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

## 4.4 Hyperlinks

Een term die je ongetwijfeld weleens voorbij hebt horen komen is *link*, of voluit benoemd: *hyperlink*. Zoals de naam al aangeeft is een link een koppeling. Waartussen? Dat is in veel gevallen tussen twee verschillende webpagina's. Maar een link kan ook verwijzen naar een ander soort media, zoals: videoclips, audio fragmenten, plaatjes, uitvoerbare bestanden, tekst bestanden, enzovoorts. Zelfs naar een specifiek stuk inhoud binnen dezelfde webpagina.

Allereerst zullen we een eenvoudige hyperlink bekijken: een link naar een bestand. Een hyperlink wordt gemaakt met het container element a. Dit element dient ook weer afgesloten te worden met een gelijknamige tag met slash ervoor: </a>. Tussen de tags komt de inhoud te staan waarop de gebruiker kan klikken (en die we in het geval van tekst vaak als onderstreept terugzien).

Met behulp van de attributen kunnen we onder andere aangeven welk bestand we gaan openen bij het klikken op de link in de HTML-pagina.

Een term die je ongetwijfeld weleens voorbij hebt horen komen is *link*, of voluit benoemd: *hyperlink*. Zoals de naam al aangeeft is een link een koppeling. Waartussen? Dat is in veel gevallen tussen twee verschillende webpagina's. Maar een link kan ook verwijzen naar een ander soort media, zoals: videoclips, audio fragmenten, plaatjes, uitvoerbare bestanden, tekst bestanden, enzovoorts. Zelfs naar een specifiek stuk inhoud binnen dezelfde webpagina.

Allereerst zullen we een eenvoudige hyperlink bekijken: een link naar een bestand. Een hyperlink wordt gemaakt met het container element a. Dit element dient ook weer afgesloten te worden met een gelijknamige tag met slash ervoor: </a>. Tussen de tags komt de inhoud te staan waarop de gebruiker kan klikken (en die we in het geval van tekst vaak als onderstreept terugzien).

Met behulp van de attributen kunnen we onder andere aangeven welk bestand we gaan openen bij het klikken op de link in de HTML-pagina.

### 4.4.1 Het href attribuut

Hier wordt net als bij afbeeldingen (dus het img element) een pad verwacht. Dit pad mag ook hier relatief of absoluut zijn.

Als waarde van de href kan in plaats van een bestandsnaam ook *mailto:emailadres* worden gebruikt. Wanneer op een hyperlink met een mailto-URL wordt geklikt, opent een browser die deze mogelijkheid ondersteunt, het e-mailprogramma en plaats het e-mailadres in het To (of Aan)-veld.

Het gebruik van een e-mail adres in het href attribuut van een <a> tag wordt sterk afgeraden. Dagelijks worden webpagina's door vele automatische zoekmachines doorlopen (onder andere) zoekende naar e-mail adressen die in de code beschreven staan. Deze kunnen dan weer gebruikt worden voor spam doeleinden.

Wil je veilig een e-mail sturen na het klikken op een link? Zorg dan dat de code die ervoor zorgt dat we een e-mail kunnen sturen geschreven is een een zogeheten server-side programmeertaal zoals PHP of ASP, waardoor de code achter de schermen ook echt achter de schermen blijft voor malafide zoekmachines.

Andere attributen van het a element worden voor een ander doel gebruikt, dit zullen we later terugzien.

```
<body>
    <a href="http://www.oneofakinds.nl"> Dit is de link
naar een webpagina</a>
</body>
```

### 4.4.2 Het target attribuut

Omdat wij na het klikken op een link niet altijd onze huidige pagina willen verlaten, bestaat er binnen HTML de mogelijkheid om een doel venster op te geven.

Target is het attribuut dat wij hiervoor gebruiken. Dit attribuut kan één van de volgende waarden bevatten:

- '_self'. Dit is de default waarde als wij het target attribuut weg zouden laten en zorgt er dus voor dat de doelpagina in dit venster, binnen dit tabblad getoond wordt. De huidige pagina verdwijnt dan uit beeld.
- '_parent'. Indien wij momenteel in een zogeheten frame zitten binnen een ander frame, dan kunnen wij aangeven met de _parent waarde aangeven dat de doelinhoud in het omliggende frame moet worden getoond.
- '_blank'. Wellicht de meest gebruikte waarde. Dit zorgt voor een nieuw venster (/popup) waardoor onze huidige pagina gewoon behouden blijft.
- '_top'. Opent het doel in het meest bovenliggende venster van deze webpagina.
- 'framenaam'. We mogen een naam van een frame opgeven waar het doel in dient te verschijnen. Frames worden verderop in dit hoofdstuk kort behandeld. Er wordt hierbij géén underscore vooraf opgegeven.

Nu we weten hoe we een link naar een bestand kunnen maken, gaan we een hyperlink naar een specifiek punt binnen een bestand maken.

Om naar bepaalde punten te kunnen navigeren moeten we op die punten ankers aanmaken. De plaats waarnaar we willen verwijzen markeren we met een willekeurig element in combinatie met het attribuut id. We markeren daarmee we de locatie waar wij naartoe willen navigeren. Zo'n markering van een locatie heet dus een anker.

Het attribuut id zullen we nog even toelichten:
Het element met het attribuut id, is voor ons de markering van een plaats binnen een document, waarnaar de bestemming van een hyperlink kan verwijzen. De waarde van het attribuut is case-sensitive.

Om nu te kunnen navigeren naar een anker moeten we in een a element (de hyperlink) als waarde van het attribuut href het id van het anker opnemen, voorafgegaan door een hekje (#). Bevindt het anker zich op dezelfde pagina, dan kunnen we volstaan met attribuutwaarde #ankerid. Bevindt het anker zich echter op een ander pagina (dus in een ander bestand), dan moeten we de attribuutwaarde uitbreiden tot:

bestandsnaam#ankerid.

Het attribuut id wordt op websites nog voor veel meer doeleinden gebruikt. Dit is er maar één van. Er mogen géén elementen op dezelfde webpagina zijn met hetzelfde id!

Binnen het volgende voorbeeld bestaan drie ankers; ankers met de namen 'anker_ol', 'anker_50' en 'anker_ul'. Daarnaast is het a element een hyperlink naar het eerste anker op dezelfde pagina. Bij het uitproberen kan het beste de browser NIET schermvullend worden geopend, maar in een kleiner formaat

```
<html>
<head><title>Voorbeeld</title></head>
<body>

    <span id="anker_ol"><h2>Een genummerde
lijst</h2></span>
    <ol>
      <li> element 1</li>
      <li> element 49</li>
      <li> <span id="anker_50"><strong>element
50</strong></span></li>
      <li> element 51</li>
      <li> element 90</li>
    </ol>

  <h2>Dummy tekst:</h2>
  <div>
    Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Fusce imperdiet diam eget nulla commodo, non
efficitur nunc interdum. Etiam varius, quam dignissim
rhoncus porttitor, diam orci ullamcorper dolor, eget
fringilla purus justo a odio. Interdum et malesuada fames
ac ante ipsum primis in faucibus. Donec justo diam,
pellentesque id maximus sollicitudin, mollis nec nulla.
Duis varius arcu et felis convallis, eget finibus dolor
fringilla. Mauris interdum dolor et rutrum rhoncus.
Curabitur pharetra urna id quam dignissim faucibus.
Pellentesque ultrices eu enim nec pellentesque. Donec
eleifend consequat diam et consectetur. Nullam et semper
arcu. Sed vel laoreet sapien, convallis varius metus. Sed
pharetra viverra metus, non eleifend metus lacinia non.
Vivamus molestie lacus ut nunc suscipit, nec rutrum ipsum
porttitor. Donec elementum luctus ligula eu commodo. Nullam
leo nisi, interdum sit amet malesuada eget, commodo sed
sapien.

Duis tincidunt ullamcorper justo a facilisis. Proin
ultricies sapien a imperdiet condimentum. Donec pulvinar
metus turpis, a commodo metus laoreet sit amet. Morbi
interdum tempor sagittis. Aliquam erat volutpat. Donec
vestibulum, turpis id cursus cursus, risus lacus placerat
dolor, eu tincidunt nulla orci ut sem. Suspendisse suscipit
velit a volutpat auctor. Curabitur quis tempus justo. Donec
rutrum, eros porttitor ullamcorper euismod, felis urna
gravida ipsum, finibus posuere elit est at nulla.

Phasellus vitae cursus augue, nec convallis sapien. Mauris
faucibus interdum dolor quis facilisis. Integer cursus,
tortor et blandit convallis, purus tortor iaculis quam,
ullamcorper eleifend sem sem nec odio. Morbi dignissim
vehicula viverra. Vivamus maximus lacinia neque, at tempus
nunc tempor ut. Praesent in euismod dui, quis mattis metus.
Aliquam nec dictum arcu. Fusce nibh odio, placerat at nisi
```

```
luctus, bibendum scelerisque justo. In nec vulputate est.
Mauris quis erat pulvinar, efficitur augue nec, dignissim
ante. Maecenas sagittis tristique bibendum. Morbi tincidunt
erat eu elementum iaculis.

Praesent et vulputate nisi. Sed malesuada tortor eu tellus
efficitur, molestie scelerisque ante eleifend. Praesent
vulputate imperdiet tortor, vitae mattis lectus hendrerit
nec. In id ligula pharetra urna dictum molestie. Integer
convallis ornare commodo. Phasellus et vestibulum augue.
Quisque et mi elit. Aenean sodales at augue et aliquet.
Praesent suscipit interdum feugiat. Nulla mattis nec mi nec
bibendum. Donec a justo libero. Fusce ac tristique arcu.

Cras elit nisi, posuere quis auctor non, mollis ac lacus.
Duis maximus sagittis dictum. Suspendisse quam metus, porta
sit amet luctus non, hendrerit at urna. Praesent vitae enim
massa. Nam hendrerit nulla non dolor molestie vestibulum.
Nulla sit amet mi pellentesque, varius arcu ut, lobortis
neque. Vivamus in massa tristique metus varius posuere.
Vivamus sagittis quam ac augue scelerisque, ac pulvinar
tellus bibendum. Cras vulputate pretium aliquam. Ut varius
eu orci ac semper. Morbi ac odio at ante vulputate
tincidunt et eget erat. Cras ac consectetur eros. Fusce
condimentum luctus bibendum. Aenean nec velit sit amet
tellus tincidunt interdum ut a sem.
  </div>

    <span id="anker_ul"><h2>Een ongeordende
lijst</h2></span>

    <ul>
      <li> een element</li>
      <li> nog een element</li>
    </ul>

    <a href="#anker_ol">Naar het begin van de pagina</span>
</body>

</html>
```

We hebben een tweede voorbeeld uitgetypt waarmee je vanuit een ander bestand naar deze ankers zou kunnen navigeren:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Dit is weer een voorbeeld van ankers en
hyperlinks</title>
  </head>
  <body>
    <a href="ankers_hyperlinks.html#anker_ol">Link naar de
genummerde lijst</a> <br/>
    <a href="ankers_hyperlinks.html#anker_50">Link naar
element50</a><br/>
    <a href="ankers_hyperlinks.html#anker_ul">Link naar de
ongeordende lijst</a><br/><br/>
    <a href="ankers_hyperlinks.html#anker_x">Link naar
niet-bestaand anker</a><br/>
  </body>
</html>
```

Merk op dat de laatste hyperlink verwijst naar een anker dat niet bestaat. De pagina wordt toch geopend en er wordt naar het begin van de pagina genavigeerd.

## 4.5 Embedding en iframes

Middels embedding kunnen externe applicaties, zoals een Adobe Flash bestand, een audio file of een video file uitgevoerd worden binnen de webpagina.

### 4.5.1 Embed element

Embedding kan zowel middels het element met de <embed> tag als middels de <audio> en <video> tags. Nog niet zo lang geleden  werd in plaats hiervan het element object gebruikt. De syntax van object is echter een stuk complexer. Embed is wat veelzijdiger dan het audio of video element omdat deze beide (en meer!) soorten bronnen ondersteunt. Het nadeel van het embed element, is dat er relatief weinig attributen mee te geven zijn om het element naar wens klaar te zetten.

Het embed element heeft een viertal mogelijke attributen, waarvan enkel het src attribuut nodig is om te kunnen functioneren. Om geheel W3C compliant te werken, is echter ook een type attribuut vereist. Daar dient het media type (video, audio, flash) opgegeven te worden. Deze typen moeten overeenkomen met vast gelegde 'media types'. Een overzicht van deze typen is te vinden op: http://www.iana.org/assignments/media-types/media-types.xhtml.

```
<html>
<head><title>Voorbeeld</title></head>
<body>
   <embed
src="https://archive.org/download/testmp3testfile/mpthreetest.mp3"
/>
</body>
</html>
```

Zowel embedded videobestanden als audiobestanden zullen direct afspelen bij het openen van een webpagina. Hiertoe zijn de autoplay="false"  attribuut en waarde bedacht. Deze werkt echter niet in Chrome of FireFox.

Er zijn work-arounds door bijvoorbeeld een Windows Media Player plugin te gebruiken.

Ons advies: gebruik het audio of video element die in hoofdstuk 8 worden besproken.

Probeer, indien mogelijk online een video of audio bestand te vinden. Sla de URL hiervan op en pas deze zelfstandig toe in een webpagina met <embed> element (bijvoorbeeld in het vorige voorbeeld).

Een voorbeeld van twee vrij te gebruiken bestanden zijn:
Audio: MP3 muziekbestand :
https://ia802508.us.archive.org/5/items/testmp3testfile/mpthreetest.mp3
Video:  MP4 videobestand:
https://ia601407.us.archive.org/32/items/ChronoTrigger_456/ChronoTrigger_456_part01_512kb.mp4
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

### 4.5.2 Frames

Tot en met HTML 4.01 werden ook 'frames' met enige regelmaat gebruikt om externe inhoud (zoals vanaf een andere server of een ander bestand) in te laden binnen de HTML pagina. Ook konden webpagina's hiermee zichtbaar opgedeeld worden in logische structuren zoals een menu, een header, een footer en een zijbalk.

Echter, hét grote nadeel van frames was dat ieder frame een apart HTML document was. Dat hield dus in, dat in dit genoemde voorbeeld, er vijf(!) verschillende webpagina's ingeladen moesten worden; vier plus de hoofdpagina waar alles in geladen wordt.

Een ander nadeel van frames was dat ieder frame met dikke randen omlijnd was, waardoor het geheel bepaald niet esthetisch was.

Als laatste speelde de beheerbaarheid een grote rol in het verval van frames. Ze waren niet goed te onderhouden en amper goed op te maken. Tegelijkertijd was CSS code in opkomst. CSS maakte het mogelijk om met gebruik van bijvoorbeeld div elementen een pagina vele malen mooier op te maken en dynamischer te (onder)houden dan een pagina die bestond uit frames.

### 4.5.3 Iframes

Er is nog één nuttig element overgebleven uit het frame framework, en dat is het zogeheten iframe. Dit element kan middels CSS code qua grootte en positie in het document, aangepast worden. Dit maakt hem een stuk flexibeler inzetbaar dan een normaal frame.

Waar en waarvoor worden iframes nu nog gebruikt? Een iframe is een goed middel om externe data, zoals reclame, te tonen op een website. Het laden van een iframe kan tegelijk met laden van andere zaken op de webpagina, dat kan schelen. Helaas blokkeert een iframe ook een zogeheten 'onload' proces. Daardoor kan het alsnog lang duren voordat de essentiële zaken van uw website zichtbaar zijn. Hier zijn weer allerlei workarounds voor bekend, die te ver gaan om in deze cursus te behandelen.

Bekijk het boek: Web Performance Tuning (O'reilly) voor meer informatie hierover: https://books.google.nl/books?id=D-rJCgAAQBAJ&lpg=PP2&ots=rRJ2Y4oriu&dq=978-0-596-00172-8&hl=nl&pg=PP1#v=onepage&q=978-0-596-00172-8&f=false.

Dit boek is ook (deels) hiernaast direct te lezen door gebruik te maken van een frame. De achterliggende code voor het iframe hiernaast is:

```
<html>
<head><title>Voorbeeld</title></head>

    <body>
      dit is een gevuld iframe
<iframe frameborder="0" scrolling="no" style="border: 0px;"
              src="https://books.google.nl/books?id=D-
rJCgAAQBAJ&lpg=PP2&ots=rRJ2Y4oriu&dq=978-0-596-00172-
8&hl=nl&pg=PP1&output=embed" width="500" height="500">
</iframe>

    </body>
</html>
```

Expert tip:
Een iFrame kan voor security lekken zorgen. Zorg er bijvoorbeeld daarom voor dat de pagina die in je iFrame geladen wordt, geen toegang kan krijgen tot het 'parent' element (je eigen webpagina). Je wilt natuurlijk cross site scripting (https://nl.wikipedia.org/wiki/Cross-site_scripting) zien te voorkomen! Lees hier (https://www.w3schools.com/tags/att_iframe_sandbox.asp) hoe je gebruik kunt maken van het nieuwe 'sandbox' attribuut, om je iFrame veiliger te maken.

Andersom wil je soms niet dat jouw eigen website als iFrame binnen een pagina ingeladen wordt. Hiervoor dient een serverinstelling aangepast te worden. De HTTP response header genaamd 'X-Frame-Option' kan ingesteld worden op 'deny' of 'sameorigin'.

'Deny' betekend: niemand kan deze webpagina in een iFrame laden. Zelfs binnen deze domeinnaam. De instelling 'sameorigin' geeft de mogelijkheid deze pagina in een iFrame te embedden wanneer het iFrame op hetzelfde domein draait als deze webpagina.

### 4.5.4 dialog

Met de <dialog> tag kunnen wij eenvoudig vensters  als het ware pop-ups maken op onze webpagina.  De inhoud is weer andere html-code. Hiervoor was voorheen vrij veel code nodig, met dit element is dit eenvoudig te doen. Enkel de positionering en opmaak dient nog te gebeuren middels CSS.
Het dialog element wordt getoond als het attribuut open is opgenomen, zonder dat attribuut blijft het verborgen. Hierdoor kan eenvoudig met javascript dit hele stuk aan en uit worden gezet.

Kijk voor meer informatie en een (beknopt) voorbeeld op MDN:
https://developer.mozilla.org/nl/docs/Web/HTML/Element/dialog.

## 4.6 Samenvatting

De toepassing van afbeeldingen en links op een webpagina is voor ons niet meer onbekend. Wij kunnen afbeeldingen toevoegen en nuttige attributen eraan toevoegen. Daarnaast is getoond welke semantische elementen belangrijk zijn bij het gebruik van afbeeldingen.

Ook hebben wij geleerd hoe wij een hyperlink maken en dat wij daarmee kunnen verwijzen naar pagina's, maar ook naar een specifiek punt in de huidige of een andere webpagina. Wij weten hoe wij ervoor kunnen kiezen het doel van de link te tonen in het huidige venster, of er nieuw venster voor te openen. Tot slot hebben wij gezien hoe wij een audiovisueel bestand kunnen embedden en wat het nut is van iframes.

# 5 Tabellen, lijsten en speciale karakters

## 5.1 Inleiding

Tabellen zijn een belangrijk aspect als het gaat om het structureren van de tekst. In dit hoofdstuk gaan we kijken hoe we tabellen kunnen aanmaken en structureren.

- Kunnen benoemen van de verschillende onderdelen van een tabel
- Het kunnen maken van een eenvoudige tabel met kolommen en rijen
- Bekend zijn met lijsten en hoe deze toe te passen zijn

## 5.2 Tabel onderdelen

Veel soorten informatie worden het best weergegeven in een structuur van rijen en kolommen. In HTML worden daar tabellen voor gebruikt. Tabellen worden opgebouwd uit verschillende elementen, die onderstaand worden besproken.

In voorgaande versies van HTML werden tabellen ook gebruikt om onderdelen van een pagina te positioneren. Inmiddels wordt dit gebruik van tabellen afgeraden, omdat Cascading Stylesheets (CSS) daar beter geschikt voor zijn.

Met het table element wordt het begin en het einde van een tabel gedefinieerd. De <table> tag dient ook weer met </table> afgesloten te worden.

Een tabel bestaat uit rijen en kolommen en boven elke kolom kunnen we een kolomkop plaatsen. Het vullen van een tabel vindt altijd plaats met behulp van rijen. Tabelrijen worden met het element tr aangemaakt. Het begin respectievelijk einde van één rij wordt aangegeven met de volgende tags: <tr> en </tr>.

Binnen elke rij hebben we de keuze uit twee elementen om data in de rijen te plaatsen: Een kolomkop geven wij aan met de begin- en sluittag <th> en </th>. Voor een gewone kolom gebruiken wij respectievelijk <td> en </td>.

De tekst binnen een th element wordt standaard vet en gecentreerd weergegeven. De tekst binnen het td element wordt standaard normaal en links uitgelijnd weergegeven. Dit kan uiteraard worden aangepast.

Aan de hand van een voorbeeld wordt duidelijk hoe de exacte structuur van een tabel er in HTML-code uitziet.
Hieronder volgt een simpel voorbeeld:

```
<html>
  <head>
    <title>Dit is een voorbeeld van een tabel</title>
  </head>
  <body  >
       <table>
      <tr>
        <th>Kolomkop 1</th>
        <th>Kolomkop 2</th>
        <th>Kolomkop 3</th>
      </tr>
      <tr>
        <td>Inhoud van cel A1</td>
        <td>Inhoud van cel A2</td>
        <td>Inhoud van cel A3</td>
      </tr>
      <tr>
        <td>Inhoud van cel B1</td>
        <td>Inhoud van cel B2</td>
        <td>Inhoud van cel B3</td>
      </tr>
    </table>
  </body>
</html>
```

Bij tabellen wordt vaak een bijschrift geplaatst. Dit kunnen we doen met behulp van het caption element.
De tag <caption> wordt altijd na de tag <table> en vóór de tag <tr> geplaatst. Ook wordt deze tag altijd afgesloten met een gelijknamige </caption> tag.

Meer gangbaar is het gebruik van een 'echte' kop zoals een <h2> element. Om deze boven meerdere kolommen te zien te krijgen, dient deze in een <td> element in de rij (<tr>) erboven geplaatst te worden. Ook moet dan het colspan attribuut ingesteld zijn. Daar komen we later in de hoofdstuk nog op terug.

Het opmaken van tabellen heeft enige voeten in aarde. Het grootste deel van een tabel kan opgemaakt worden met behulp van CSS code. Onderstaand kunt u kijken hoe wij dit kunnen doen. Verderop in de cursus gaan wij tabellen opdelen in groepen, die vervolgens ook weer op te maken zijn.

```
<html>
  <head>
    <title>Dit is een voorbeeld van een tabel</title>
      <STYLE>
         table {
             border-spacing 0px;
               }
         td{
             border: solid 1px red;
               color: blue
```

```
        }
      </STYLE>
  </head>
  <body  > ......(rest als vorige voorbeeld)
```

Wij gaan nu zelf een tabel maken voor een veilinghuis. Dit veilinghuis verkoopt allerlei producten (van kunst en antiek tot auto's en gereedschap). De informatie die in de tabel komt te staan zijn producten die op een veiling verkocht gaan worden.

Maak een nieuw HTML bestand met daarin een nieuwe tabel. Bedenk zelf wat voor nuttige kolommen we kunnen maken en kies er minimaal vier. Zorg ervoor dat er nette kopjes boven iedere kolom staan én vul de tabel direct met in ieder geval drie rijen.

Je hoeft verder zelf nog niets aan de opmaak te doen. Indien je al een enige CSS programmeer ervaring hebt, mag je uiteraard proberen wat opmaak op de tabel toe te passen.
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

### 5.3 Tekst over kolommen of rijen verspreiden

Opeenvolgende cellen kunnen zowel in horizontale als in verticale richting worden samengevoegd. Dit kan bijvoorbeeld van pas komen wanneer een kolomkop over meerdere kolommen verspreid moet worden.

Voor het samenvoegen in horizontale richting gebruiken we attribuut colspan van het td (of th) element.  Colspan geeft aan hoeveel opeenvolgende cellen in horizontale richting samengevoegd moeten worden tot één cel. Het attribuut wordt geplaatst in de eerste van de samen te voegen cellen.

Voor het samenvoegen in verticale richting gebruiken wij het rowspan attribuut. Rowspan geeft aan hoeveel opeenvolgende cellen in verticale richting samengevoegd moeten worden tot één cel. Het attribuut wordt geplaatst in de eerste van de samen te voegen cellen.
Het stuk code in het style (CSS) blok hoeft u niet te kennen. Dit is wederom CSS code, enkel gebruikt om het effect van de colspan en rowspan te verduidelijken.

```
<html>
  <head>
    <title>Dit is een voorbeeld van een tabel</title>
      <style> td {
                        border: solid 1px black;
            }
</style>
  </head>
  <body  >
       <table>
  <caption> Dit is het bijschrift bij de tabel </caption>
  <tr>
    <th>Kolomkop 1</th>
```

```
    <th>Kolomkop 2</th>
    <th>Kolomkop 3</th>
    <th>Kolomkop 4</th>
  </tr>
  <tr>
    <td colspan="2">inhoud van cel A1</td>
    <td>Inhoud van cel C1</td>
    <td>Inhoud van cel D1</td>
  </tr>
  <tr>
    <td rowspan="2">inhoud van cel A2</td>
    <td>Inhoud van cel B2</td>
    <td>Inhoud van cel C2</td>
  </tr>
  <tr>
    <td>Inhoud van cel B3</td>
    <td>Inhoud van cel C3</td>
    <td>Inhoud van cel D3</td>
  </tr>
</table>
  </body>
</html>
```

De pagina ziet er als volgt uit:

Dit is het bijschrift bij de tabel

| Kolomkop 1 | Kolomkop 2 | Kolomkop 3 | Kolomkop 4 |
|---|---|---|---|
| inhoud van cel A1 | | Inhoud van cel C1 | Inhoud van cel D1 |
| inhoud van cel A2 | Inhoud van cel B2 | Inhoud van cel C2 | |
| | Inhoud van cel B3 | Inhoud van cel C3 | Inhoud van cel D3 |

We kunnen het volgende constateren:

- Cel A1 neemt 2 kolommen (colspan="2") in beslag, zodat in de eerste rij de vier kolommen allemaal gevuld worden.
- In de tweede rij met data worden maar 3 kolommen gevuld, omdat elke cel in 1 kolom past.
- In de derde rij zijn alle kolommen gevuld; de eerste kolom is gevuld door A2 en de andere kolommen zijn gevuld met de cellen B3, C3 en D3. Dit is het gevolg van de samenvoeging van de eerste td in het voorlaatste tr element (rowspan="2").

### 5.4 Eigenschappen groeperen

We hebben de mogelijkheid om de rijen of kolommen van een tabel te groeperen, zodat we voor meerdere cellen tezamen overeenkomende eigenschappen kunnen vastleggen.

### 5.4.1 Kolomgroepen

Met het element colgroup kunnen we binnen een tabel een aantal kolommen groeperen. De gewenste kenmerken kunnen we opgeven als attribuut bij dit element en deze gelden dan uiteraard voor alle kolommen van de groep.

Het <colgroup> element heeft ook weer een gelijknamig sluitelement.

Het enige in HTML 5 bruikbare attribuut is: span. Deze geeft aan hoeveel naast elkaar gelegen kolommen tot de kolomgroep behoren (default waarde is 1).

In het volgende voorbeeld zal het één en ander worden verduidelijkt.

```
<html>
  <head>
    <title>Dit is een voorbeeld van een tabel</title>
  </head>
  <body>
      <table border="2">
            <colgroup style="width: 300px; background:
lightgray">
            </colgroup>
            <colgroup span="3" style="width: 100px;
background: lightblue">
            </colgroup>
            <tr>
              <td height="40">cel a1</td>
              <td>cel b1</td>
              <td>cel c1</td>
              <td>cel d1</td>
              <td>cel e1</td>
            </tr>
            <tr>
              <td height="40">cel a2</td>
              <td>cel b2</td>
              <td>cel c2</td>
              <td>cel d2</td>
              <td>cel e2</td>
            </tr>
            <tr>
              <td height="40">cel a3</td>
              <td>cel b3</td>
              <td>cel c3</td>
              <td>cel d3</td>
              <td>cel e3</td>
            </tr>
      </table>
</body>
</html>
```

### 5.4.2 Rijgroepen

Net als kolommen, kunnen we rijen met dezelfde eigenschappen groeperen. Het groeperen kunnen we toepassen op de header, de footer of op rijen in de body-sectie. Hiervoor maken we respectievelijk gebruik van de elementen thead, tfoot en tbody (de cellen in de rijen die behoren tot de header en de footer bevatten informatie over de kolommen, de cellen in de rijen die behoren tot een body bevatten de tabelgegevens).

Deze elementen hebben de volgende open en sluit tags:
<thead> en </thead>
<tbody> en </tbody>
<tfoot> en </tfoot>

Het mooie van deze rijgroepen is dat wij ook hieraan met CSS code een opmaak kunnen toekennen.

```
<html>
  <head>
    <title>Dit is een voorbeeld van een tabel</title>
  </head>
  <body>
      <table>
            <thead style="color: red; background:
lightblue;">
              <tr>
                  <td>Kop cel 1</td>
                  <td>Kop cel 2</td>
                  <td>Kop cel 3</td>
              </tr>
            </thead>
            <tbody>
              <tr>
                  <td>Inhoud rij 1 cel 1</td>
                  <td>Inhoud rij 1 cel 2</td>
                  <td>Inhoud rij 1 cel 3</td>
              </tr>
              <tr>
                  <td>Inhoud rij 2 cel 1</td>
                  <td>Inhoud rij 2 cel 2</td>
                  <td>Inhoud rij 2 cel 3</td>
              </tr>
            </tbody>
            <tfoot style="color: blue; background:
lightgray;">
              <tr>
                  <td>Voet cel 1</td>
                  <td>Voet cel 2</td>
                  <td>Voet cel 3</td>
              </tr>
            </tfoot>
      </table>
</body>
</html>
```

In dit voorbeeld wordt een tabel met drie rijen en vijf kolommen getoond. Door in het eerste colgroup element geen attribuut span en bij het tweede element wel een attribuut span met waarde '3' op te nemen, vormt de eerste kolom een aparte kolomgroep en vormen de drie daarop volgende kolommen ook één kolomgroep. De laatste kolom valt niet binnen een kolomgroep. We zien verder dat de opgegeven eigenschappen steeds voor de gehele groep gelden.

De achtergrondkleur en de breedte van de groepen wordt met behulp van stylesheets via het attribuut ‘style’ meegegeven, omdat deze eigenschappen geen attributen (meer) zijn van het element colgroup.

Zo ziet bovenstaande tabel er nu uit:

| | | | | |
|---|---|---|---|---|
| cel a1 | cel b1 | cel c1 | cel d1 | cel e1 |
| cel a2 | cel b2 | cel c2 | cel d2 | cel e2 |
| cel a3 | cel b3 | cel c3 | cel d3 | cel e3 |

Van oudsher hebben we nog de mogelijkheid om binnen een kolomgroep enkele kolommen een afwijkende opmaak te geven; met het col element kunnen we kolomgroepen verdelen in kleinere groepen. Dit element heeft dezelfde attributen als het colgroup element.

Dit <col> element heeft eveneens een gelijknamige sluittag </col>.

Het col element valt binnen een colgroup element. De eigenschappen die we bij het colgroup element opgeven gelden ook voor de col elementen. Wanneer een eigenschap bij zowel colgroup als col is opgegeven, geldt de bij col opgegeven waarde.

Het col element wordt niet meer ondersteund in de nieuwste standaar

## 5.5 Lijsten

Gegevens kunnen we in opsommingen weergeven met behulp van de elementen ol en ul. Zij creëren respectievelijk een geordende en een ongeordende lijst.

Het ul element (Unordered List) definieert het begin en einde van een ongeordende lijst, waarvan de lijstelementen voorafgegaan worden door een symbool (bv: een vierkantje, een rondje). De list wordt geopend en gesloten met respectievelijk de <ul> en </ul> tags.

Middels het optionele type attribuut kunnen wij aangeven welk symbool er voor de opsomming gebruikt wordt. Mogelijke waarden zijn: disc (dicht rondje), circle (open rondje) en square (vierkantje).

Het ol element (Ordered List) definieert het begin en einde van een geordende lijst, waarvan de lijstelementen voorafgegaan worden door een volgnummer (bv: 1,2,3 of a,b,c). Deze list wordt geopend en gesloten met de <ol> en </ol> tags.

Bij deze lijst geeft het type attribuut ook aan welk symbool er voor de opsomming gebruikt wordt. Mogelijke waarden zijn: 1 (1,2,3), a (a,b,c), A (A,B,C), i (i,ii,iii), I (I,II,III). Het bijbehorende optionele start attribuut geeft de startwaarde (een cijfer) aan van het eerste item in de list. Wanneer dit attribuut de waarde 3 heeft en het type van de list is i, dan zal de lijst met iii beginnen.

Elke list zal gevuld worden met een lijst van één of meerdere gegevens. Het li element (List Item) definieert een list item van een ongeordende (ul) of een geordende (ol) lijst. Het li element heeft ook gelijknamige open <li> en sluit </li> tags.

Het attribuut type heeft bij dit element een andere definitie. Hij bepaalt namelijk welke markering voor het betreffende item wordt gebruikt, wanneer deze moet afwijken van het type dat voor het ul of het ol element is gedefinieerd. Voor de waarde kan men kiezen uit alle waarden van het attribuut type bij ol of ul elementen.

Elk ul en ol element bevat dus één of meer li elementen. Lijsten kunnen genest worden: in elk li element kunnen één of meer ul of ol elementen opgenomen zijn. Indien de tekst langer is dan er op één regel past, wordt deze op de volgende regels ingesprongen weergegeven.

```
<html>
  <head>
    <title>Dit is een voorbeeld van een tabel</title>
  </head>
 <body>
        Voorbeeld van een ongeordende lijst met rassen uit
een spel:<br/>
        <ul>
            <li>Mens</li>
            <li>Gnoom</li>
            <li>Orc</li>
            <li>Elf</li>


              </ul>

        Voorbeeld van een geordende lijst met klassen uit
een spel:<br/>
        <ol>
            <li>Magi&#235;r</li>
            <li>Dru&#239;de</li>
            <li>Jager</li>
            <li>Strijder</li>
        </ol>
</body>
</html>
```

tel nu zelf een lijst samen van bijvoorbeeld uw eigen hobby's, interesses, sport of bijvoorbeeld spelers uit je favoriete voetbalclub. Het is goed om hiervoor een nieuw HTML document te schrijven. Uiteraard mag je ook bovenstaande voorbeeld gebruiken, maar aangeraden wordt om 'from scratch' te beginnen. Deze oefening is extra, mochten er niet uitkomen neem contact op met de docent.

In de praktijk wordt een lijst vooral veel gebruikt om een navigatie menu mee te definiëren. Je kunt hiermee namelijk de structuur van een opsomming beschrijven en ernaar verwijzen met CSS code of JavaScript code om het geheel dynamisch te maken.

Omdat dit belangrijk is om te herkennen, volgt onderstaand nog een voorbeeld.

```
<html>
  <head>
    <title>Dit is een voorbeeld van een tabel</title>
      <style>
      li {
  display: inline;
}

ul {
  padding:0px;
}</style>
  </head>
 <body>
        Voorbeeld van een geordende lijst met klassen uit
een spel, horizontaal weergegeven met behulp van CSS.<br/>

        <ul>
              <li>Magi&#235;r</li>
              <li>Dru&#239;de</li>
              <li>Jager</li>
              <li>Strijder</li>
        </ul>
</body>
</html>
```

### 5.5.1 Definitielijsten

Om gegevens gestructureerd weer te geven kan gebruik worden gemaakt van definitielijsten. Een definitielijst is een lijst van termen (ook wel trefwoorden genoemd). Vervolgens wordt dit betreffende trefwoord uitgelegd. Een definitielijst wordt aangemaakt met het dl element (Definition List). De bijbehorende tags zijn: <dl> en </dl>.

Een list wordt binnen het element dl verder opgebouwd met de elementen dt en dd. Ook dezen hebben gelijknamige begin tags en eind tags.

Het dt element (definition table) definieert een term in een definitielijst, het dd element (definition description) definieert de beschrijving.

```
<html>
  <head>
    <title>Dit is een voorbeeld </title>
  </head>
 <body>
    <span>Een voorbeeld met snelwegen:</span>
```

```
    <dl compact>
      <dt>A2</dt>
        <dd>Snelweg tussen Amsterdam en Maastricht</dd>
      <dt>A59</dt>
        <dd>Snelweg tussen Breda en Eindhoven</dd>
    </dl>
    <span>Een voorbeeld met twee 5hart-cursussen:</span>
<br/>
    <dl>
      <dt>Cursus html</dt>
        <dd> Met de komst van het web is grondige kennis
van HTML onontbeerlijk. </dd>
      <dt>Cursus javascript</dt>
        <dd>Javascript is een logische vervolgtraining op
de cursus "Bouwen van webpagina's met HTML". </dd>
    </dl>
</body>
</html>
```

Het uitlijnen van de tekst (links de term, rechts de omschrijving) kan natuurlijk goed met tabellen óf met behulp van CSS code.

## 5.6 Speciale tekens

In HTML hebben we de mogelijkheid om speciale tekens (character entities) op te nemen. Dit zijn tekens zoals accenten op klinkers, maar ook bijvoorbeeld de tekens die in HTML eigenlijk dienen om bijvoorbeeld het begin (<) of het eind (>) van een tag aangeven. In moderne browsers gaat het weergeven van tekens vaak wel goed, maar in oudere browsers is het gebruik van character entities een must.
Hieronder vinden we een aantal voorbeelden. Op W3schools.com een uitgebreide lijst van entities en een lijst van speciale tekens opgenomen. Resp.
https://www.w3schools.com/html/html_entities.asp en
https://www.w3schools.com/html/html_symbols.asp

| | | | | | |
|---|---|---|---|---|---|
| < | < | > | > | " | &quot ; |
| & | & | § | § | è | è |
| ÷ | ÷ | × | × | à | à |
| é | é | ë | ë | ï | ï |
| â | â | á | á | | |
| ö | ö | ü | ü | | |

```
<html>
  <head>
    <title>Dit is een voorbeeld </title>
  </head>
 <body>
    Een element start altijd met een tag. <br/>
```

```
    Een tag start met &lt; en eindigt met &gt;. <br/>
    Een attribuutwaarde wordt tussen &quot;dubbele
quotes&quot; opgenomen. <br/>
</body>
</html>
```

### 5.7 Samenvatting

Wij kunnen vanaf nu zelfstandig tabellen maken, bestaande uit rijen (tr) en kolommen (td). Deze kunnen we eventueel groeperen met colgroups, of overlappende kolommen definieren met het span attribuut. Kopjes zijn toe te passen en ook een onderverderling van een tabel in een header, body en footer is nu bekend. Een ander belangrijk onderwerp was het gebruik van lijsten. (on)geordende lijsten en het gebruik ervan in een menu. Tot slot zijn ook definitielijsten en het gebruik van bijzondere tekens voorbij gekomen.

# 6 Formulieren

## 6.1 Inleiding

Formulieren (forms) worden gebruikt om de gebruiker (vaak de bezoeker van een pagina op een website), de mogelijkheid te bieden informatie in te voeren en te verzenden. Voorbeelden van formulieren zijn: aanmeldingen, enquêtes, bestellingen, of veelgestelde vragen. Voor formulieren worden speciale elementen gebruikt waarmee we onder andere invoervelden, keuzevelden, selectielijsten en knoppen kunnen aanmaken. De belangrijkste elementen voor het maken van formulieren zullen wij in dit hoofdstuk behandelen.

**Voordat u begint**
Vanaf dit hoofdstuk zullen wij (wederom) met regelmaat gebruik maken van <style> tags in onze voorbeelden. De code die daarbinnen gebruikt wordt hoeft u voor deze cursus niet te (her)kennen of beheersen. Deze code is enkel gebruikt om sommige voorbeelden te verduidelijken qua opmaak.

Daarnaast: in dit hoofdstuk zullen de elementen fieldset en legend gebruikt worden om elementen te groeperen. De uitleg van deze twee elementen komt aan het einde van dit hoofdstuk aan bod.

- Weten waar een HTML formulier toe kan dienen
- Toe kunnen passen van gangbare formulier onderdelen (input, textarea, select)
- Kunnen benoemen van nieuwe HTML 5 elementen
- HTML 5 attributen kunnen gebruiken op formulier elementen

Ga naar de Vijfhart webpagina en blader naar het cursus zoekformulier
Bekijk de paginabron en ga op zoek naar het element form. Probeer stukken code die eronder staan te relateren aan het resultaat.
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

## 6.2 Element FORM

Als eerste kijken we naar het element form; dit element definieert het begin en het einde van een formulier. Het element kan vier attributen bevatten, waarvan twee belangrijke attributen bepalen op welke wijze de informatie uit het formulier verzonden moet worden en waarheen.

Het element form en zijn attributen worden hieronder toegelicht. Het form element zelf heeft de gelijknamige open en sluit tag <form> en </form>.

Een kort overzicht van de belangrijkste attributen van het form element:

method — Geeft aan op welke wijze de informatie uit het formulier verzonden moet worden. Mogelijke waarden zijn 'get' en 'post'. Hierbij maakt de 'get' methode gebruik van verzending van data met behulp van de adresbalk. Data wordt met 'get' ongecodeerd verzonden. Dat kán natuurlijk voor veiligheidsproblemen zorgen. Wél is de input eenvoudig af te handelen om dat alle invoervelden

vermeld zullen staan in de adresbalk. 'Post' is in veel gevallen te aan te raden verzend methode.

action Geeft aan waar de informatie uit het formulier heen gestuurd moet worden. De bestemming is meestal een script of programma op de server dat de informatie uit het formulier moet verwerken. De informatie van het formulier kan ook zonder tussenkomst van een script of programma direct naar een opgegeven e-mailadres worden gestuurd.

autocomplete Kan de waarde "on" of "off" bevatten. Dit betekent of het formulier qua velden automatisch aangevuld dient te worden indien een gebruiker bepaalde waarden invult (waarden worden in de browser vast gehouden).

novalidate De aanwezigheid van dit attribuut (zonder waarde) betekent dat dit formulier niet middels HTML browser validatie gevalideerd wordt, wanneer hij verzonden wordt naar de server.

De attributen method en action zijn de belangrijke attributen in dit verhaal. Om het formulier verwerkt te laten worden op de server, moet minimaal het action attribuut worden toegevoegd. Op het moment dat de gegevens verstuurd worden, geeft action aan welk script op de server de gegevens moet verwerken. Dit versturen vindt plaats middels een zogeheten submit event. Deze kan plaats vinden met behulp van JavaScript code, maar in deze cursus gebeurt dit middels een knop. Hierover later meer.

De code: <form action=”test.php”> zorgt ervoor dat de file ‘test.php’ de gegevens verwerkt.

Zoals je inmiddels weet, werken we in deze cursus enkel op een client machine. Alles wat we immers nodig hebben is Notepad en de Internet browser. Het is belangrijk dat we inzien dat er pas interactie is met de server op het moment dat gegevens worden verzonden. Het laten verwerken van de gegevens van een HTML-pagina behoort tot een vervolgcursus. Vandaar dat we ook niet dieper ingaan op de bovenstaande attributen. Wel behandelen we hier de elementen die ervoor zorgen dat gegevens kunnen worden verwerkt. Het element form is er één van.

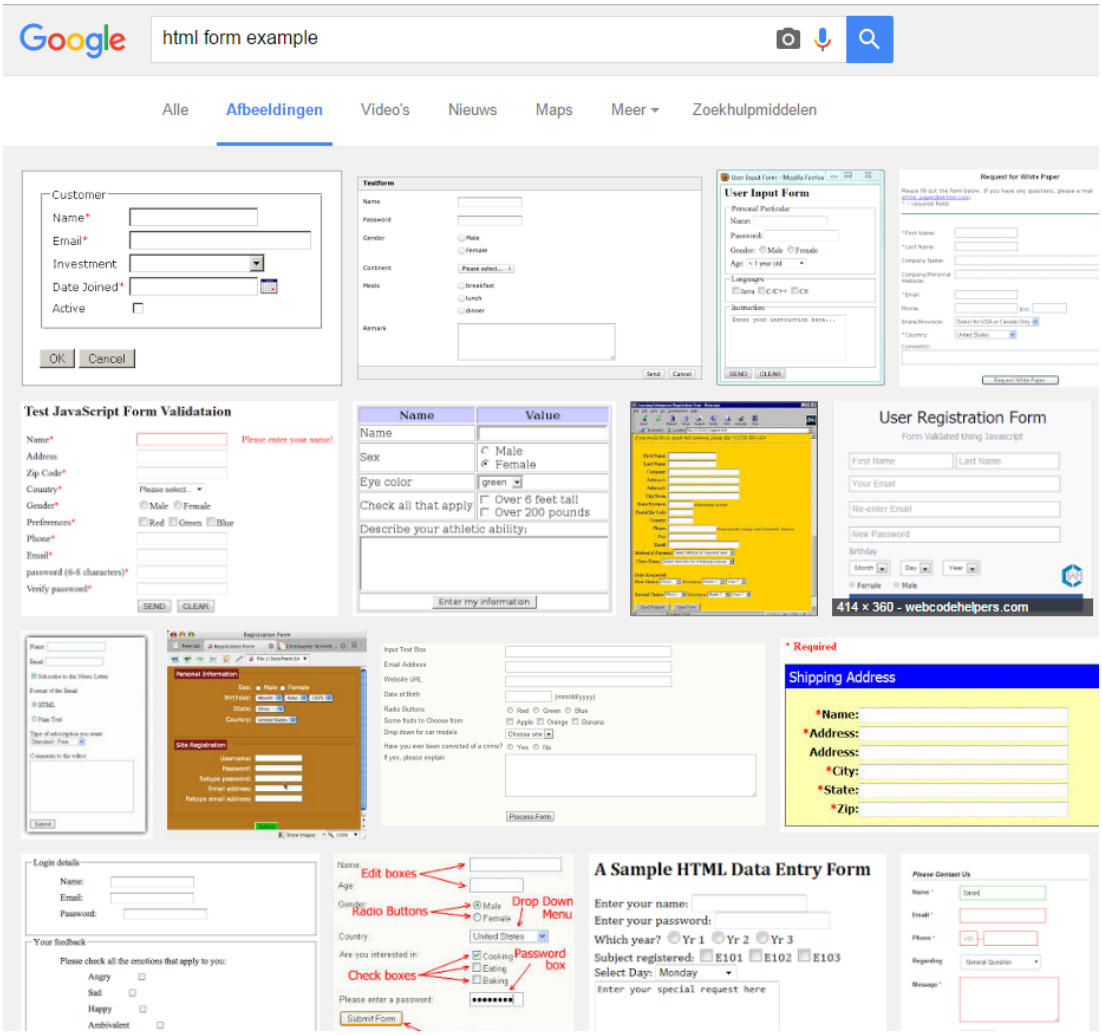
Het form element wordt toegepast in combinatie met (onder andere) één of meer van de volgende elementen: input, textarea en select. Elk element mag meerdere malen gebruikt worden. Deze elementen worden in het vervolg van deze paragraaf besproken. Voor de verwerking van het formulier moet bij elk element het name attribuut worden meegegeven. Bij het versturen van gegevens naar de server kunnen we alleen gegevens zenden door te verwijzen naar de naam van de elementen binnen het formulier. Het attribuut name zal bij de afzonderlijke elementen niet genoemd worden, omdat het toch voor elk element dezelfde betekenis heeft.

Naast de drie genoemde elementen input, textarea en select kan een formulier natuurlijk nog steeds andere elementen bevatten die we in de loop van de cursus zijn tegengekomen (tekst, paragrafen, tabellen, lijsten, et cetera). Nu gaat het er alleen om dat we de gegevens ook kunnen verzenden en daarvoor hebben we de tag <form> nodig.

Binnen één document kunnen meerdere formulieren opgenomen worden. Ze moeten echter wel na elkaar geplaatst worden; het opnemen van een formulier binnen een ander formulier is niet toegestaan.

Wanneer je zoekt in Google images met het zoekwoord 'html forms', zal je een aantal voorbeelden van formulieren zien:

# 6 Formulieren




Je ziet dat er formulieren in vele vormen en maten zijn. Omdat je zélf de regie in handen hebt als het gaat om de vormgeving ervan, kunt je ervoor zorgen dat het formulier er naar wens uit ziet door CSS code te gebruiken.

De objecten binnen een form worden onafhankelijk van elkaar geïnterpreteerd en uitgelijnd. Met behulp van tabellen kan je bijvoorbeeld de tekst en de velden mooi in twee kolommen uitlijnen. Ook kunt u CSS stijlen gebruiken om het geheel uit te lijnen.

## 6.3 Textarea

Het textarea element is een tekstvak waarin de gebruiker over meerdere regels tekst in kan voeren. Dit element heeft een begin én eindtag <textarea> en </textarea>. Ertussen mag tekst komen te staan die wij in dit tekstveld willen zien.

Textarea (cols="90" rows="4")

```
It began with the forging of the great rings.
Three were given to the elves, immortal,
wisest and fairest of all beings.
Seven to the Dwarf Lords,
```

Een tweetal belangrijke attributen:

rows Bepaalt hoe hoog het tekstvak is, uitgedrukt in het aantal rijen (regels) tekst. De hoogte heeft betrekking op de weergave van het tekstvak en niet op de hoeveelheid regels die een gebruiker kan invoeren.

cols Bepaalt hoe breed het tekstvak is, uitgedrukt in het aantal kolommen (karakters) tekst. De breedte heeft betrekking op de weergave van het tekstvak en niet op de hoeveelheid karakters die een gebruiker op een regel kan invoeren. De tekst in een tekstvak wordt weergegeven in een lettertype met een vaste letterafstand.

Met een ander attribuut zoals wrap kunnen meer functionaliteiten van het element worden gespecificeerd.

In het textarea element moeten minimaal de attributen rows en cols worden opgenomen. Als optie kan tussen de begintag en eindtag van het textarea element een standaard tekst geplaatst worden, welke vervolgens in het tekstvak wordt weergegeven.

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
<body>
    <form>
            <fieldset>
                  <legend>Textarea (cols="90"
rows="4")</legend>
                  <textarea cols="90" rows="4">
                        It began with the forging of the
great rings.
                        Three were given to the elves,
immortal,
                        wisest and fairest of all beings.
                        Seven to the Dwarf Lords,
                        great miners and craftsmen of the
mountain halls.
                        And nine, nine rings were gifted
to the race of men,
                        who above all else desired power.
                  </textarea>
            </fieldset>
      </form>
</body>
</html>
```

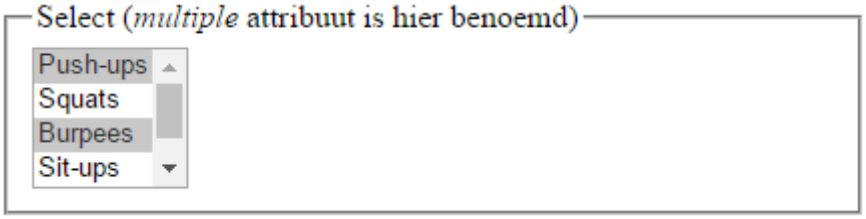
### 6.4 Keuzelijsten elementen SELECT en OPTION

Uitschuif keuzelijsten worden in formulieren veelvuldig gebruikt. Een lijst neemt in het document slechts beperkte ruimte in, maar kan toch een groot aantal keuzemogelijkheden bevatten. Met het select element geven we aan dat we een lijst gaan aanmaken. Per option element geven we een afzonderlijke lijstwaarde. De begin en sluit tag van een select element zijn respectievelijk <select> en </select>. We kiezen voor het select element omdat we graag een keuzelijst willen aanbieden. Daarom is het ook logisch daar meerdere opties

(keuzes) in te vermelden. Deze keuzelijst kunnen wij vullen met HTML (zie onderstaand voorbeeld), maar bijvoorbeeld ook met Javascript of een server-side programmeertaal zoals PHP of ASP die voor ons de HTML code genereert.

Enkele belangrijke attributen van het select element zijn:

multiple — Geeft aan of uit keuzelijst meerdere keuzen tegelijk geselecteerd kunnen worden. Dit attribuut heeft geen waarde. Dus enkel het benoemen ervan is voldoende. Hoe ziet dit eruit?




Size — Geeft aan hoeveel keuzemogelijkheden van een meervoudige keuzelijst zichtbaar moeten zijn. Omdat een schuifbalk aanwezig is, kan het aantal te selecteren keuzemogelijkheden groter zijn dan het aantal zichtbare.

Binnen het select element horen één of meer option elementen te zijn gemaakt. Deze hebben ook als open en sluit tag <option> en </option>.
Het option element heeft tweetal attributen, waarvan value de meest belangrijke is:

selected — Geeft aan welke optie standaard geselecteerd dient te worden. De eerste keuzemogelijkheid in een keuzelijst wordt geselecteerd wanneer nergens deze eigenschap opgenomen is. Dit attribuut heeft geen waarde!

value — Bepaalt welke waarde naar de server gestuurd moet worden als de keuze wordt geselecteerd (gecombineerd met de waarde van het name attribuut van het select element). Zonder attribuut value wordt de keuze zelf (de inhoud van het element option) naar de server gestuurd. Dit is dus een belangrijk attribuut voor een <option> tag.

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
 <body>
  <form>
        <fieldset>
        <legend>Select (<em>multiple</em> attribuut is hier
benoemd)</legend>
        <select name="oefeningen" multiple>
            <option selected value="1">Push-ups</option>
            <option value="2">Squats</option>
            <option selected value="3">Burpees</option>
            <option value="4">Sit-ups</option>
            <option value="5">Pull-ups</option>
        </select>
  </form>
</body>
</html>
```

In dit geval worden 'push-ups' en 'burpees' geselecteerd. Dit zijn meerderen en dat kan in dit geval, gezien wij het attribuut 'multiple' hebben opgenomen in onze <select> tag.

# 6 Formulieren

Maak nu zelf een select element aan op een webpagina. Voeg hieraan een lijstje met uw favoriete muzikanten of sportclubs aan toe. Ieder item weer binnen losse <option> tags uiteraard.

Probeer het tot slot ook mogelijk te maken dat een eindgebruiker meer dan één optie kan kiezen uit jouw select element.
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

## 6.5 Element INPUT

Met dit element wordt een veld gedefinieerd, waarin de gebruiker gegevens kan invoeren. Hierbij kan het gaan om verschillende vormen, zoals:

- tekstgegevens (al of niet verborgen)
- een selectie met behulp van keuzerondjes en aankruisvakjes
- opdrachten tot het verzenden van de ingevoerde of geselecteerde informatie
- het herstellen van de veldwaarden naar de defaults
- kleuren kiezer
- range kiezer
- datum kiezer
- bestand kiezer

Het input element heeft geen sluit tag, dus enkel de tag: <input/>.

Dit element wordt ontzettend veel gebruikt en heeft een behoorlijk aantal attributen, zoals: value, readonly, disabled, size en maxlength. Sinds HTML 5 zijn er nog eens een behoorlijk aantal bijgekomen. Bekijk welke attributen dit zijn via de website van MDN. Een groot deel van deze attributen komen verderop in dit hoofdstuk uitgebreid aan bod.

Het belangrijkste attribuut genaamd type bepaalt wat voor soort gegevens het invoerveld accepteert. We zullen de waarden hier opsommen, in de volgende subparagrafen worden ze uitvoeriger besproken. Van oudsher zijn de volgende waarden mogelijk: text, password, checkbox, radio, submit, reset, button, image, hidden, file.

De standaardwaarde voor dit attribuut is text.

Het element heeft meer attributen dan die hierboven zijn beschreven. Afhankelijk van het type hebben de meeste attributen een andere functionaliteit. Vandaar dat dit bij de beschrijving van het betreffende type verderop terug te vinden is.

### 6.5.1 Tekstveld, type attribuut waarde: text

Dit element definieert een veld waar we één woord of een enkele regel tekst kunnen invoeren.

Input type = "text" (dat is de default waarde)

Voornaam:

Achternaam:

Leeftijd:

# 6 Formulieren

Dit zijn de meest gebruikte velden binnen een formulier. Denk aan een veld waar men zijn of haar naam, woonplaats, e-mail adres, enzovoorts kan invullen.

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
 <body>
  <form>
            <fieldset>
                  <legend>Input type = "text" (dat is de
default waarde)</legend>
                  <table>
                        <tr>
                              <td>Voornaam:</td>
                              <td><input type="text" /></td>
                        </tr>
                        <tr>
                              <td>Achternaam:</td>
                              <td><input type="text" /></td>
                        </tr>
                        <tr>
                              <td>Leeftijd:</td>
                              <td><input type="text" /></td>
                        </tr>
                  </table>
            </fieldset>
      </form>
</body>
</html>
```

Maak een formulier, from scratch (dus geen copy-paste werk). Als basis om naar te kijken mag gerust het voorgaande voorbeeld gebruikt worden, maar probeer dit zoveel mogelijk zelfstandig op te bouwen.

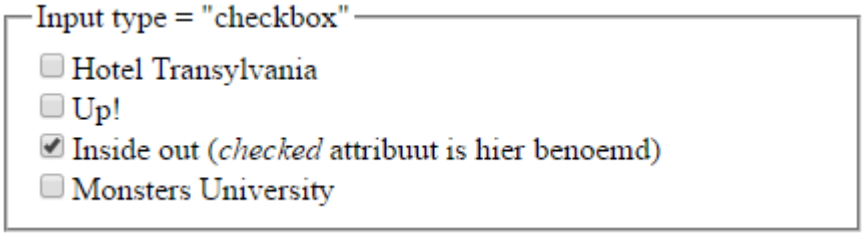
Kies ook een thema waar je dit formulier voor wilt maken; uitnodiging voor een personeelsfeest, antwoord formulier voor een puzzelwedstrijd of bijvoorbeeld het aanmeld formulier voor de nieuwbrief van je eigen webpagina.

Het formulier dient meerdere <input> tags te gebruiken. Eventueel mogen ook een <select> en <textarea> gebruikt worden. Plaats het geheel binnen de juiste formulier tags om het te omsluiten en fictief te kunnen versturen.

Opmaak hoeven wij voor nu nog niet veel rekening mee te houden. Een enkele <br/> na ieder invoerveld zou al prima zijn.
Wil je toch al het nodige realiseren qua opmaak? Gebruik dan een tabel en/of kijk dan naar het voorgaande voorbeeld.
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

### 6.5.2 Aankruisvakje, type attribuut waarde: checkbox

Dit element definieert een aankruisvakje. De gebruiker kan het vakje uit- of aanvinken.




Een checkbox wordt vaak gebruik in combinatie met de volgende attributen:

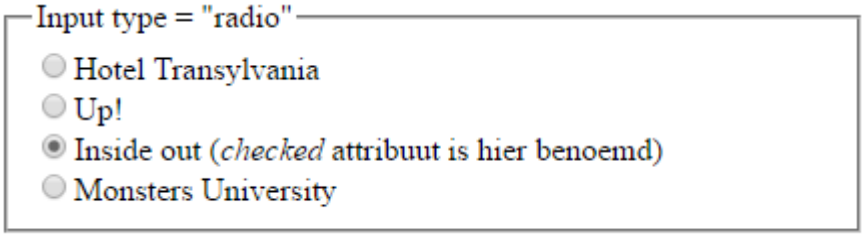
Value Geeft de waarde die naar de server gestuurd moet worden als het vakje geselecteerd is.

Checked Geeft de standaardkeuze. Dit attribuut heeft geen waarde!

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
 <body>
    <form>
            <fieldset>
                  <legend>Input type = "checkbox"</legend>
                  <input type="checkbox" />Hotel
Transylvania<br/>
                  <input type="checkbox" />Up!<br/>
                  <input type="checkbox" checked />Inside
out (<em>checked</em> attribuut is hier benoemd)<br/>
                  <input type="checkbox" />Monsters
University<br/>
            </fieldset>
      </form>
  </body>
</html>
```

### 6.5.3 Keuzerondjes, type attribuut waarde: radio

Dit element definieert een keuzerondje dat deel uitmaakt van een set van keuzerondjes. De gebruiker kan één van deze opties uit de set selecteren.




De bij een keuzerondje horende attributen zijn:

Value Geeft de waarde die naar de server gestuurd moet worden.

Name Geeft de naam van de set radiovelden.

Bij elkaar horende velden dienen dezelfde naam te krijgen!

Checked Geeft de standaardkeuze. Dit attribuut heeft geen waarde!

Radiogroepen passen we toe als het aantal keuzemogelijkheden niet al te groot is. Bij meer dan drie à vier keuzemogelijkheden is het beter gebruik te maken van een lijst, op te bouwen met elementen select en option (begin van dit hoofdstuk behandeld). Hieronder volgt ander voorbeeld om uit te proberen.

```
Input type = "radio"
Uitstekend
Zeer goed
Goed (checked attribuut is hier benoemd)
Voldoende
Matig
Slecht
```

Wanneer u deze HTML draait ziet het er goed uit. Maar bij uitproberen zul u zien dat u alle radiovelden onafhankelijk van elkaar aan en uit kunt zetten. Wanneer u wilt dat ze één groep vormen moet u elk input element voorzien van het attribuut name met dezelfde waarde.

Welk type element zou de beste keuze zijn als wij de gebruiker zijn of haar geslacht willen laten kiezen? Een checkbox of een radio?
Is de keuze niet helder? Probeer het dan uit! Maak een geslacht keuzeveld op basis van een checkbox én op basis van een radio.

Kennen we nog een ander soort element dat voor bovenstaande geschikt zou zijn?
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

### 6.5.4 Verzendknop, type attribuut waarde: submit

Dit element definieert een knop, die er voor zorgt dat de gegevens uit het formulier naar de webserver worden verstuurd.

Het attribuut *value* geeft de waarde die op de knop wordt geplaatst (default waarde is: 'Submit Query').

De combinatie van deze submit button en de waarde van het attribuut action van het form element (hierin wordt aangegeven welke pagina de gegevens moet gaan verwerken), zorgt pas voor een bepaalde actie op de server zoals het aanmaken van een gebruiker, het plaatsen van een blog item of registreren voor een nieuwsbrief. Het method attribuut van de <form> tag, geeft aan op welke manier wij willen communiceren met de server. In geval van 'post', zullen alle ingevulde formuliervelden gecodeerd naar de server worden verzonden. In geval van de waarde 'get' in het method attribuut is dat niet het geval. Alle elementen met bijbehorende waarden zullen dan als URL naar de server gestuurd worden, die ook de eindgebruiker kan zien. Ongecodeerd, dus ook niet veilig. Wél snel echter. Wij zullen hier verder in deze cursus niet op ingaan.

Elk formulier kan één submit-knop bevatten. Dit is ook de reden dat er soms meerdere formulieren op een pagina te vinden zijn. De gegevens van een bestelformulier en een aanmeldingsformulier die zich op dezelfde pagina bevinden hebben bijvoorbeeld niets met elkaar te maken en moeten onafhankelijk van elkaar kunnen worden verstuurd.

### 6.5.5 Herstelknop, type attribuut waarde: reset

Dit element definieert een knop, die de beginwaarden van de verschillende controls toont (en dus ook alle ingevoerde tekst wist).

Het attribuut value gebruiken wij (ook) bij dit type input om de tekst van de knop aan te geven.

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
<body>
  <form action="niet_relevant">
      <h1>Dit is een voorbeeld formulier </h1>
        tekstveld: <input type="text" value="typ hier iets"
/><br/>
        tekstveld: <input type="text" value="en druk op
'herstel'" /><br/>
      herstellen : <input type="reset"
value="herstel"/><br/>
      zenden : <input type="submit" value="zenden"/><br/>
  </form>
</body>
</html>
```

### 6.5.6 Verborgen tekst, type attribuut waarde: password

Dit element definieert een veld waarin de gebruiker een gevoelige tekst, zoals wachtwoorden, in moet voeren. De karakters in het veld worden vaak met een asterisk (*) weergegeven, maar een ander karakter komt ook voor. Voor dit type element gelden dezelfde attributen als voor het type text.

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
<body>
    <hr/>
    <form action="niet_relevant">
      <h1> Dit is een voorbeeld formulier </h1>
      wachtwoord : <input type="password" name="wachtwoord"
value="geheim" /><br/>
    </form>
</body>
</html>
```

### 6.5.7 Bestand selectie veld, type attribuut waarde: file

Met dit element kunnen wij een bestand uitzoeken, die wordt meegegeven aan het formulier en aan de server zodra wij het formulier verzenden. Het element met type file heeft nog een extra attribuut genaamd: multiple. Dit betekent dat een gebruiker meerdere bestanden kan selecteren.

Probeer het onderstaande voorbeeld uit om te zien welk effect het file attribuut heeft.

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
<body>
  <form>
            <fieldset>
                  <legend>Input type = "file"</legend>
                  Selecteer een bestand:<br/>
                  <input type="file" />
                  <br/><br/>
                  Selecteer meerdere bestanden:<br/>
                  <input type="file" multiple /><br/><br/>
                  <em>(meerdere bestanden selecteert u door
de &lt;ctrl&gt; toets ingedrukt te houden
                  bij het kiezen van de bestanden)</em>
            </fieldset>
      </form>
</body>
</html>
```

## 6.6 Gegevens groeperen

Vaak is het mooi om bepaalde gegevens binnen een formulier te groeperen. Hiervoor kunnen we gebruik maken van de elementen fieldset, legend en label.

### 6.6.1 Elementen FIELDSET en LEGEND

Het fieldset element kan gebruikt worden om een aantal gerelateerde controls van een formulier te groeperen. Het zorgt ervoor dat er een kader om wordt geplaatst. Dit element heeft geen attributen en wordt geopend en gesloten met de volgende tags: <fieldset> en </fieldset>.

Het legend element zorgt ervoor dat er een bijschrift bij de door fieldset gegroepeerde controls wordt weergegeven. De bijbehorende tags zijn: <legend> en </legend>.

Let op: Wanneer je het legend element wil gebruiken moet dit het eerst element binnen fieldset zijn. Daarna volgen de andere elementen. Het fieldset element mag weer een nieuw fieldset element bevatten.

### 6.6.2 Element LABEL

Er is ook een element waarmee wij kunnen aangeven welk stukje beschrijving hoort bij welk element. Tekst die vlak voor of vlak na een element staat, hoeft natuurlijk niet perse ook bij dát element te horen. Hoe geven wij dat aan? Bekijk het voorbeeld hieronder. Kijk naar het element label met het attribuut genaamd for.

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
<body>
  <form>
      <fieldset>
            <legend><strong>Geslacht</strong></legend>
            <label for="naamveld">Naam</label>
            <input type="text" name="naamveld" /><br/>
            <input type="radio" name="geslacht" value="M"
checked />Man <br/>
            <input type="radio" name="geslacht"
value="V"/>Vrouw <br/><br/>
            <input type="checkbox" id="chkVoorwaarden" />
            <label for="chkVoorwaarden">[Ik ga
akkoord]</label>
      </fieldset>
  </form>
</body>
</html>
```

Is het opgevallen dat je kunt klikken op de tekst (het label element) naast het checkbox element? Dat is de kracht van een label bij een input type="checkbox" element. Vooral omdat je vaak niet precies op het vakje kunt of zult klikken met de muis.

Het is in de praktijk slimmer om bij een <label> tag te verwijzen naar het id van een element, in plaats van een name van een element (zie bovenstaand).

Probeer te bedenken waarom.

Hint: een id is binnen de webwereld veel toepasbaarder dan een name attribuut. Denk aan JavaScript en CSS code die ook naar elementen met een bepaald id kunnen verwijzen. Vroeger diende er verwezen te worden naar het name attribuut, inmiddels wordt dat afgeraden.;

## 6.7 HTML 5 Elementen

Vanaf HTML 5 zijn er een aantal element typen bij gekomen die onder andere te gebruiken zijn voor cijfers, data en overige speciale typen. Wij zullen deze typen onderstaand gaan bespreken.

### 6.7.1 Elementen voor cijfers

Hierbij kunnen wij denken aan een invoer velden met type ´number´, ´range´ en ´tel´.

# 6 Formulieren

Bekijk onderstaand voorbeeld en probeer te achterhalen wat ieder element precies doet. Wij behandelen in dit voorbeeld niet alle typen voor getallen. Deze kan je nalezen op: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input.

```
<html>
 <head>
  <title>Dit is een voorbeeld formulier </title>
 </head>
<body>
 <form>
  <fieldset>
   <legend>Input type = text/number/range</legend>
   <table>
    <tr>
     <td>
      <label for="leeftijd_tekst">Leeftijd (tekst):</label>
     </td>
     <td>
      <input type="text" id="leeftijd_tekst" />
     </td>
    </tr>
    <tr>
     <td>
       <label for="leeftijd_ nummer">Leeftijd
(nummer):</label>
     </td>
     <td>
       <input type="number" id="leeftijd_nummer" />
     </td>
    </tr>
    <tr>
     <td>
      <label for="leeftijd_nummer_bereik">Leeftijd (nummer
met bereik 0-100):</label>
     </td>
     <td>
      <input type="number" min="0" max="100"
id="leeftijd_nummer_bereik" />
     </td>
    </tr>
    <tr>
     <td>
      <label for="leeftijd_range">Leeftijd (nummer met
bereik 0-100):</label>
     </td>
     <td>
      <input type="range" min="0" max="100" step="20"
id="leeftijd_range" />
      </td>
     </tr>
    </table>
   </fieldset>
  </form>
```

```
  </body>
</html>
```

Is het opgevallen dat wij gewoon een getal kleiner dan 0, of groter dan 100 kunnen invullen in het nummerveld, ondanks dat er een min en max attribuut bij het nummerveld is gebruikt? Dat komt doordat er geen fysieke invoercontrole op dit veld zit, maar enkel het bereik van de waarde in het veld wordt ingesteld voor het geval wij gebruik maken van de bijbehorende pijltjes (omhoog en omlaag).

Is het daarnaast opgevallen dat er bij het range type element geen waarden onder vermeld staan? Dat is geen standaard functionaliteit en kan eventueel met Javascript code opgelost worden.

Tot slot heb je ongetwijfeld gemerkt dat het gebruik van labels, maar óók het gebruik van een tabel om een formulier op te maken het nodige extra typewerk kost. Dit is niet bepaald een keuze. Het alternatief is gebruik te maken van CSS code voor de opmaak, maar ook daarbij komt de nodige code kijken.

### 6.7.2 Elementen voor data

Om de gebruiker een datum uit te laten kiezen gebruiken wij één van de volgende input typen:

- date
- time
- datetime-local
- week
- month

Deze input typen spreken erg voor zich wanneer wij de code hiervan zouden zien.

Maak zelf een webpagina met een formulier. Maak in dat formulier gebruik van minimaal 3 van de voorgaande input typen.

Zorg dat de eindgebruiker gemakkelijk in kan zien wat er in deze velden kan ingevoerd kan worden.
Deze oefening is extra, mocht je er niet uitkomen, neem dan contact op met de docent.

### 6.7.3 Overige input typen

Met de komst van HTML 5, zijn er een hoop zaken vereenvoudigd. Wilden wij voorheen graag een kleurkeuze veld maken, waarbij men zelf de kleuren kon uitkiezen, dan was dat een hoop programmeerwerk met JavaScript en CSS code.  Ook het maken van een veld waar enkel een e-mail adres ingevuld mag worden, diende met JavaScript uitgebreid gecontroleerd te worden.

HTML 5 biedt de uitkomst!

Onderstaande voorbeeld maakt gebruikt van de elementen:

- datalist
- color (input)
- email (input)
- search (input)
- url (input)

Kijk wat er gebeurt als je een ongeldig e-mail adres invult en wat er gebeurt bij het zoekveld, indien wij zoeken op de eerste letter(s) van de opgegeven productnamen.

```
<html>
 <head>
  <title>Dit is een voorbeeld formulier </title>
 </head>
<body>
 <form>
  <fieldset>
   <legend>Persoonlijke enquete</legend>
   <table>
    <tr>
     <td>
      <label for="product">Wat is het leukste product: <br/>
      <em>keuze uit: Mahonie..., Noten..., Ebben...,
Palisander</em></label>
     </td>
     <td>
      <datalist id="productenlijst">
       <option>Mahonie bureau met ingelegd bestempeld leer</option>
       <option>Noten wortelhout secretaire</option>
       <option>Ebben fiches kistje</option>
       <option>Palisander 70's salontafel</option>
      </datalist>
      <input type="search" list="productenlijst" id="product" />
            <select list="productenlijst">
            </select>
     </td>
    </tr>
    <tr>
     <td>
      <label for="kleur">Favoriete kleur:</label>
     </td>
     <td>
      <input type="color" id="kleur" /></br>
      <input type="text" id="kleurcode" value="#000000"  disabled
/>
     </td>
    </tr>
    <tr>
     <td>
      <label for="website">Favoriete website:</label>
     </td>
     <td>
      <input type="url" id="website" />
     </td>
    </tr>
    <tr>
     <td>
      <label for="email">Contact e-mail adres:</label>
     </td>
     <td>
```

```
      <input type="email" id="email" />
     </td>
    </tr>
    <tr>
     <td colspan="2">
      <input type="submit" value="Verzend" />
     </td>
    </tr>
  </fieldset>
 </form>
<script>
var kleurveld = document.getElementById("kleur").addEventListener(
"change", function(){

document.getElementById('kleurcode').value=document.getElementById(
'kleur').value;
});

document.getElementsByTagName("form")[0].addEventListener("submit"
,function(){  alert("Gegevens zijn verzonden...");
});
</script>
</body>

</html>
```

Je hebt wellicht in dit voorbeeld gemerkt dat niet alle element typen doen wat je zou verwachten. Zo heeft het input element met het type search eigenlijk enkel een semantische betekenis; het is een zoekveld.

De toepassing hiervan in ons voorbeeld is uitgebreid met een verwijzing naar een <datalist> tag. Is dat opgevallen? Waarvoor zorgt een verwijzing naar een <datalist>?

Het zoekveld vult automatisch aan. Maar waarmee? Waar haalt hij deze gegevens vandaan? In de praktijk worden deze gegevens opgehaald vanuit een database (lokaal of op een server). Daarbij kan als tussenstap een zogeheten datalist gebruikt worden, waarin deze waarden in HTML (onzichtbaar in de output) worden opgeslagen.

### 6.7.4 Datalist element

Wij hebben in het vorige voorbeeld voor het eerst kennis gemaakt met een datalist element. Dit is tegenwoordig een relatief veel gebruikt element. Hiermee kunnen wij een lijst van keuzemogelijkheden opslaan en 'later' gebruiken binnen dezelfde webpagina.

Dit element begint en sluit respectievelijk met de <datalist> en </datalist> tags. Hij heeft ook één of meerdere elementen in zich (sub elementen) die <option> heten en ook weer worden afgesloten met </option>. Denk maar aan een select element, maar dan zonder enige visuele weergave.

Een praktisch voorbeeld:

Op de webpagina van een bakkerij wil een klant graag een saucijzenbroodje bestellen, alleen weet meneer niet hoe hij dat woord precies schrijft. Het liefst typt hij dus in het zoekveld op die pagina in: 's' .... en de rest weet hij nog niet. Wat is er nou mooier dan dat er direct een aantal mogelijke opties worden weergegeven?

Vanuit performance oogpunt kan het gebruik van een datalist ook voordelen hebben. De mogelijke waarden kunnen bijvoorbeeld bij het inladen van de pagina worden gevuld met waarden uit een database. Hierdoor hoeft er niet tussentijds met de database verbinding gemaakt te worden om deze waarden op te halen. Uiteraard zijn er hier meerdere wegen naar Rome, maar dit kán dus een voordeel opleveren.

Maak zelf een HTML webpagina met daarop een zoekveld die gevuld wordt met mogelijkheden uit een datalist element. Gebruik als inspiratie bijvoorbeeld een lijst van uw hobby's, een lijst met favoriete voetbalclub's, streaming televisie series, games, computer onderdelen of bijvoorbeeld een lijst van verschillende soorten vloerbedekking.

Maak vervolgens ook een select element (dropdown menu), met daarin dezelfde waarden.
Voeg tot slot, als tweede alternatief, een lijst met radio input items toe met dezelfde waarden als bovenstaande lijsten.

Zijn er verschillen tussen deze lijsten? Zo ja welke? Zo nee, zijn er wel verschillen te maken qua functionaliteit?
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

### 6.7.5 Progress element

Met behulp van de <progress> tag kunnen wij een element maken dat als progressiemeter dienst doet. Wij kunnen hiermee bijvoorbeeld uitstekend de voortgang van een proces tonen. Er is op het element een (start) waarde ('value') in te stellen, en een maximale ('max') waarde middels de bijbehorende attributen.

In de praktijk zien wij een progressiebalk (vaak rijkelijk) opgemaakt met CSS code.

Onderstaand volgt een voorbeeld van een progressiebalk die wij met behulp van JavaScript vullen.

```
<html>
  <head>
    <title>Dit is een voorbeeld formulier </title>
  </head>
<body>
    <main>
        <button id="start_btn">Start</button><br/>
        <span id="introtekst"></span><br/>
        <progress id="balk" value="0.0"></progress><br/>
    </main>
      <script>
```

```
                  // De volgende code hoeft u niet te begrijpen en
wordt in onze Javascript cursus behandeld:
            var balk;

    window.onload = function(){

      document.getElementById("start_btn").addEventListener("click",
start, false);

      balk = document.getElementById("balk");
            }

            function start()
            {
              balk.value = 0.0;
              document.getElementById("start_btn").disabled =
"disabled";
              var tekst = document.getElementById("introtekst");
              tekst.innerHTML = "Er wordt nu een 'zware' (lees:
tijdverdrijvende) taak uitgevoerd";
              var klaar = false;
              console.log("loop");
              setInterval(function() {
                        console.log(balk.value);
                        if (balk.value!=1.0)
                        {
                              balk.value += 0.1;
                        }
                        else{
                              return;
                        }
                  }, 1000);

                  document.getElementById("start_btn").disabled =
"";
            }
      </script>
</body>
</html>
```

Er zijn nog tal van opmaak mogelijkheden beschikbaar voor het stijlen van een progress element middels CSS. Denk aan een ander soort kader, een kleurverloop in de achtergrond of in de balk, of zelf een plaatje als achtergrond van het geheel.

### 6.7.6 Meter element

Vaak wordt het meter element gebruikt als alternatief voor een progress element. Daar is hij semantisch niet voor bedoeld. De <meter> tag is bedoelt om te gebruiken om een (vaste) waarde aan te geven binnen een bepaald bereik (bijvoorbeeld: 0.5l van het pak melk van 1.5l gevuld, of: 800gb vrij op de harddisk van 4tb).

Dit element heeft meer mogelijke attributen dan het progress element, namelijk:

- value
- min
- max
- low
- high
- optimum
- form

Hierbij is enkel het value attribuut verplicht. De meter moet een waarde hebben. De overige attributen geven het element meer functionaliteit. Kijk wat deze attributen doen op de website van MDN en probeer ze te begrijpen.

Maak een formulier met daarop een drietal meter elementen. Dus voeg een <meter> tag toe voor de volgende meetwaarden:
Drukmeter fietspomp (2 bar van de 12 bar)
Snelheidsmeter (80km/h van de 200km/h)
Weegschaal (100gram van de 1kg)
Test ook in je output of de meters goed zijn ingesteld.
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

### 6.7.7 Overige formulier elementen

Het keygen element hebben wij hier bewust niet besproken omdat deze deprecated is, ofwel: het wordt aangeraden dit element niet meer te gebruiken.

## 6.8 HTML 5 Elementen attributen

Onderstaand volgt een overzicht met alle overige HTML 5 attributen die op form elementen van toepassing zijn. Later volgen er nog enkele attributen die alleen op een <form> element toepasbaar zijn.

- disabled
- maxlength
- readonly
- size
- value
- autocomplete
- autofocus
- form
- formaction
- height
- width
- list
- min
- max
- multiple
- pattern (regexp)
- placeholder
- required
- step

Het form attribuut is een vreemde eend. Hiermee kunnen wij bij een element aangeven dat deze bij een ander formulier hoort wanneer dat formulier verzonden wordt. Dit was in HTML 4 nog niet mogelijk. Daarnaast is het formaction attribuut ook een vreemde in deze lijst. Dit attribuut mag enkel toegepast worden op een <input type="submit" /> of een <button

type="submit"> worden toegepast. Men kan hiermee de standaard actie van het formulier overschrijven (action attribuut van de <form> tag).

Bekijk onderstaand voorbeeld om de andere meest gebruikte attributen te begrijpen.

```
<html>
 <head>
  <title>Dit is een voorbeeld formulier </title>
 </head>
<body>
 <form>
  <table>
   <tr>
    <td>Disabled</td>
    <td><input type="text" disabled /></td>
   </tr>
   <tr>
     <td>Maxlength</td>
    <td><input type="text" maxlength="4" /></td>
   </tr>
   <tr>
    <td>Readonly</td>
    <td><input type="text" readonly /></td>
   </tr>
   <tr>
    <td>Value</td>
    <td><input type="text" value="al ingevuld" /></td>
   </tr>
   <tr>
    <td>Autocomplete</td>
    <td><input type="text" autocomplete /></td>
   </tr>
   <tr>
    <td>Autofocus</td>
    <td><input type="text" autofocus /></td>
   </tr>
   <tr>
    <td>List</td>
    <td><input type="text" list="opties" /></td>
   </tr>
   <tr>
    <td>Pattern</td>
    <td><input type="text" pattern="^[0-9]{3}[-][0-9]{7}$"
title="Telefoonnummer: 012-3456789" /></td>
   </tr>
   <tr>
    <td>Placeholder</td>
    <td><input type="text" placeholder="Vul hier uw waarde
in..." /></td>
   </tr>
   <tr>
    <td>Required</td>
    <td><input type="text" required /></td>
   </tr>
```

```
    <tr>
     <td rowspan="2"><input type="submit" value="Test" />
     </td>
    </tr>
   </table>
   <datalist id="opties">
    <option>Keuze 1</option>
    <option>Keuze 2</option>
    <option>Keuze 3</option>
   </datalist>
  </form>
</body>
</html>
```

Breid nu de bovenstaande HTML uit, met minimaal twee elementen met andere attributen dan die wij in het voorbeeld hebben gebruikt (maar wél in bovengenoemde lijst staan).

Voeg daarnaast een element toe die controleert of er een geldige postcode is ingevoerd.

Tip: gebruik hiervoor het pattern attribuut en kijk op MDN voor meer informatie over reguliere expressies (onder de kop: Writing a regular expression pattern) . Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

Attributen die op een <form> tag gebruikt kunnen worden, zijn:

enctype — Hiermee kunnen wij aangeven hoe de te verzenden gegevens in een formulier gecodeerd dienen te worden. Deze optie is nuttig bij het gebruik van de waarde 'post' bij het 'method' attribuut. Dit enctype attribuut kan één van de volgende drie waarden bevatten:
*application/x-www-form-urlencoded*: dit type stuurt alle elementen met inhoud door als één grote tekst 'string'. Dit maakt het een snelle manier van verzenden, maar niet efficiënt voor binaire data (zoals files). Omdat niet alfanumerieke tekens in een %xx vorm worden doorgestuurd, waarbij xx de representatie is van het teken als ASCI character. Dit neemt altijd minimaal twee karakters in beslag en kan dus bij grote hoeveelheden binaire data inefficiënt zijn.
*multipart/form-data*: de data wordt in losse zogeheten 'MIME' messages verstuurd. Ieder bericht heeft zijn eigen eigenschappen en is het meest interessant om te gebruiken indien er grote hoeveelheden data middels dit formulier verzonden dienen te worden.
*text/plain*: de gegevens worden niet gecodeerd doorgestuurd. Snel, maar onveilig voor gevoelige informatie.

novalidate — De aanwezigheid van dit attribuut zonder waarde is al voldoende om hem aan te zetten. Als alternatief mag ook als waarde 'novalidate' ingevuld worden. Wat doet dit attribuut? Het formulier zal niet gevalideerd worden, zelfs indien er velden in zitten die wél validatie wensen.

target — Waar komt de output terecht die wij terugkrijgen van de server? Dit is standaard de waarde '_self' (let op de underscore), maar mag ook zijn '_blank' voor een nieuwe lege pagina, '_parent' voor het bovenliggende frame of '_top' voor het meest bovenliggende frame (e.v.t het huidige venster dus).

## 6.9 Samenvatting

n dit hoofdstuk hebben wij geleerd wat de basics van een webformulier zijn. Een groot deel van alle formulier elementen hebben wij gezien en mee leren werken. Daaronder vielen onder andere een groot aantal elementen met de tag input en allen met een verschillende type attribuut. Ook hebben wij gezien hoe wij elementen in een formulier kunnen groeperen en hoe wij uiteindelijk de gegevensvelden weer kunnen wissen of verzenden.

# 7 Audio en video

## 7.1 Inleiding

Zoals we al hebben gelezen kunnen wij audio-visuele elementen in HTML toevoegen met behulp van de behandelde <embed> tag. We hebben daarbij gemerkt dat dit element niet zo veelzijdig is als dat wij zouden willen wanneer het gaat om weergave van specifieke video of audio formaten, óf voor de weergave van alternatieven indien er geen browser ondersteuning is.

In dit hoofdstuk zullen wij gaan leren werken met de video en audio elementen. Wij zullen ervaren hoe wij alternatieve audio en video formaten kunnen weergeven indien een formaat niet door de browser van de gebruiker ondersteund wordt. Het draait om het creëren van een zo optimaal mogelijke gebruikerservaring!

- Kunnen maken van een video element en deze kunnen laten starten
- Het kunnen maken en gebruiken van het audio element
- Het nut van het track element kennen en dit element ook kunnen toepassen

## 7.2 Video element

Met behulp van de <video> tag kunnen wij een video zoals een opname of animatie tonen aan de webpagina bezoeker. In de <video> tag kunnen wij een aantal regels voor het weergeven van een videovenster, zoals:

- breedte en hoogte van de video
- wel of geen knoppenbalk tonen
- wel, deels of geen 'preload' van de video
- wel of niet automatisch starten van de video

Kijk voor een overzicht van de bijbehorende attributen van de <video> tag op MDN.
Let op: om een video af te laten spelen, dient één van de volgende zaken geregeld te zijn:

- autoplay dient aan te staan, óf
- knoppenbalk dient aanwezig te zijn, óf
- JavaScript code dient hem te kunnen starten

Indien de browser de <video> tag niet ondersteund, wordt tekst weergegeven die tussen de <video> en </video> tags staat.

Het is altijd verstandig om voor de zekerheid een stukje tekst op te nemen in het video element zodat de eindgebruiker in ieder geval een gebruiksvriendelijke melding krijgt indien er geen video's worden ondersteund.

```
<html>
 <head>
   <title>HTML 5 video voorbeeld</title>
 </head>
 <body>
  <video controls>
            <source src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/small_video.mp4"
type="video/mp4"/>
```

```
            Het afspelen van deze technisch Lego helicopter
video wordt door uw browser niet ondersteund.
            <img src="http://cdn3.wpbeginner.com/wp-
content/uploads/2015/08/error.jpg" />
  </video>
 </body>
</html>
```

Zoals je in dit voorbeeld hebt kunnen zien, bevat het video element hier een aantal <source> tags. Deze source elementen zijn vergelijkbaar met het source element dat wij binnen een <picture> tag kunnen gebruiken. Binnen een <video> tag is altijd minimaal één <source> tag verplicht.

In een <source> tag geven wij met het attribuut src aan waar ons bron bestand staat (relatief op de server of met link naar een video op een andere webpagina). Daarnaast mag er een type opgegeven worden om extra duidelijkheid voor de browser te creëren.

Waarom staan er meerdere <source> tags in het voorbeeld op de vorige pagina? Het is sterk aan te raden alternatieve video formaten van het af te spelen bestand op te geven middels meerdere <source> tags.

Let op: om een zo breed mogelijk scala aan browsers te ondersteunen, wordt momenteel aangeraden minimaal de volgende formaten te ondersteunen / te beschrijven met een <source> tag:

- mov
- ogv
- webm

Hoe zorgt je ervoor dat je videobestand in deze drie formaten op de server staat en dus leverbaar is aan de bezoeker? De video dient hiervoor omgezet te worden met een zogeheten video convertor. Hiervoor zijn vele tools verkrijgbaar. Het is natuurlijk zaak dat deze video's niet te groot zijn (hoge downloadtijd), maar niet té laag van kwaliteit zijn. Laat hier dus je multimedia specialist naar kijken.

Pas het bovenstaande voorbeeld zo aan, dat het huidige MP4 bestand de laatste source optie is na: mov, ogv en webm. Dus, voeg daarvoor een aantal extra sources toe.
Deze oefening is extra, mocht je er niet uitkomen neem contact op met de docent.

### 7.2.1 Poster attribuut

Stelt je eens voor, dat je op je website meerdere video's hebt staan. Om je site zo gebruikersvriendelijk als mogelijk in te richten, zouden de eerste beelden van de video's eigenlijk zó voor zich moeten spreken, dat het voor de gebruiker direct duidelijk is waar hij of zij moet klikken. Dat is niet snel en kosteloos te realiseren. Het is óók niet praktisch om boven iedere video een apart kopje te moeten plaatsen, waarmee duidelijk wordt gemaakt waar de video over gaat. Dat mág wel, maar het liefst spreekt het (eerste) videobeeld natuurlijk voor zich.

Oplossing: gebruik het poster attribuut van de <video> tag. Met dit attribuut kan een plaatje worden aangegeven wat wordt getoond als de video nog geladen wordt en nog niet gestart is.

# 7 Audio en video

Voeg aan de vorige opdracht een poster attribuut toe.

## 7.3 Audio element

Om audio weer te kunnen geven met HTML, kunnen wij het beste gebruik maken van een audio element. Met dit element kunnen wij een audiospeler genereren, zoals wij dat met de <video> tag kunnen voor een videospeler.

Ook bij dit element zien wij in de praktijk dat de speler met JavaScript en CSS code wordt opgemaakt om hem te differentiëren van de 'standaard' spelers. In deze cursus gaan wij echter niet te veel in op CSS en JavaScript code.

Google op "html 5 audio player examples" en kijk dan onder afbeeldingen

Wanneer wij het audio element met bijbehorende <audio> en </audio> tag gebruiken, dienen wij binnen dit element één of meerdere <source> tags erin te verwerken.

De <audio> tag kan, net als de <video> tag enkele attributen bevatten die iets zeggen over:

- controls - of wij wél of geen besturingselementen willen zien bij deze audiofile (aan te raden indien autoplay niet aan staat.
- autoplay - moet de audiofile direct starten zodra dit element geladen is op de webpagina?
- loop - hiermee kunnen wij aangegeven of de file herhaald moet worden weergegeven (telkens opnieuw afspeelt)
- src - in het src element zetten wij zoals altijd het pad naar de audiofile die wij willen afspelen
- preload - dient het audio bestand alvast ingeladen te worden voordat het afgespeeld kan worden?  De waarde hiervan kan ingesteld worden op: none, auto of metadata. Auto betekent dat de audio file wordt gedownload, ook al weten wij nog niet of de gebruiker hem zal gaan afspelen. De derde optie metadata staat enkel voor het ophalen van metadata zoals de speelduur en volume.

```
<html>
<head>
    <title>HTML 5 audio voorbeeld</title>
    <style>
      /* In dit 'style' blok,
         maken we het audio element op met CSS code */

      audio {
        height:       100px;
        background-color: green;
      }
    </style>
  </head>
  <body>
    <audio controls>
```

```
    <!-- Geen IE, ongecomprimeerd, beste kwaliteit -->
    <!-- uitgezet i.v.m. filesize <source
src="Zelda_SNES.wav" type="audio/wav"> -->
    <!-- Volledig ondersteund, gecomprimeerd -->
      <source src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/Zelda_SNES.mp3"
type="audio/mpeg">
    <!-- Géén IE/Safari, gecomprimeerd -->
      <source src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/Zelda_SNES.ogg"
type="audio/ogg">

      Het afspelen van deze audiofile wordt door uw browser
niet ondersteund.
      <img src="http://cdn3.wpbeginner.com/wp-
content/uploads/2015/08/error.jpg">
    </audio>

  </body>
</html>
```

In bovenstaand voorbeeld heb je kunnen zien dat wij bij de <audio> tag gelijksoortige attributen kunnen opgeven als bij de <video> tag. Ook wordt hierbij geadviseerd gebruik te maken van verschillende sources voor de audio. Er kan onderling verschil in kwaliteit en bestandsgrootte zitten.

Een overzicht van speelbare audio formaten:

- wav; ongecomprimeerd, neemt veel ruimte in beslag
- ogg; gecomprimeerd, maar ook beschikbaar in ongecomprimeerde vorm
- mp3; gecomprimeerd

Hiervan wordt het mp3 formaat door de meeste browsers ondersteund.

Zoals vaker heeft alles zijn prijs. Wil je bijvoorbeeld hoge kwaliteit audio files leveren aan je bezoekers, dan zullen deze files meer schijfruimte in beslag nemen dan audio bestanden van minder hoge kwaliteit. Dat komt door een hogere bitrate. Dat betekent dat een dergelijk bestand er langer over zal doen om gedownload te worden en afgespeeld kan worden. Compressie zorgt voor een kleinere bestandsgrootte, maar soms ook voor verlies van audio kwaliteit.

Voor meer informatie over de ondersteuning van verschillende audio formaten, verschillen tussen audio codecs en bestand extenties, bekijk de Mozilla Developer Nework webpagina, ook wel: MDN.

## 7.4 Het track element

Met de invoering van HTML 5 is het mogelijk gemaakt zowel audio als video elementen te voorzien van ondertitels (zoals onder een video opname voor een route beschrijving, of als songtekst onder een muziekstuk.

Wij gebruiken hiervoor de <track> tag. Deze sluit ook zichzelf (dus wij hoeven geen losse </track> tag te gebruiken). Qua attributen hebben wij de keuze uit:

- default
- kind
- chapters
- descriptions
- metadata
- subtitles
- label
- src
- srclang

Hierbij is enkel het src attribuut noodzakelijk om te kunnen functioneren. Het is echter goed gebruik om ook de taal (srclang) en het label wat men ziet (label) op te geven. Ook het wel of niet aanwezig maken van het attribuut default is een overweging. Dit bepaalt of deze subtitle gebruikt wordt indien de gebruiker geen eigen specifieke subtitle wilt gebruiken.

Het formaat van een subtitle bestand is 'WebVVT', met als gebruikte bestand extentie: *.vvt. Onderstaand ziet u een voorbeeld van een opgesteld subtitle bestand voor 'The Lord of the Rings'.

Wilt u meer weten over het zelfstandig kunnen maken van subtitle bestanden, bekijk dan: MDN over WebVVT.
https://developer.mozilla.org/en-US/docs/Web/API/Web_Video_Text_Tracks_Format

```
<!DOCTYPE html>
<html>
      <head>
            <title>Subtitle voor Lord of the Rings</title>
      </head>
      <body>
            <video controls>
                  <source src="lotr.mp4" type="video/mp4">
                  <source src="lotr.ogv" type="video/ogv">
                  <track src="lotr.vtt" srclang="en"
label="English">
            </video>
      </body>
</html>
```

```
Inhoud .vtt:

WEBVTT

NOTE
Op deze ondertitels van Lord of the Rings, zijn geen
rechten te ontlenen.

1
02:46:50.000 --> 02:47:00.000
- FRODO: It's gone. It's done.
- SAM: Yes Mr. Frodo. It's over now.
```

```
2
02:47:10.000 --> 02:47:20.000
- FRODO: I can see the Shire. The Brandywine River.
- Bag End. Gandalf's fireworks. The lights on the party
tree.

NOTE
Verder gaan deze ondertitels (voor alsnog) niet.
```

Let op: Dit voorbeeld kan je alleen gebruiken indien je zelf een webserver hebt waar je dit voorbeeld op kunt uitproberen. Google Chrome ondersteunt wegens veiligheidsoverwegingen niet het afspelen van lokale bestanden. Er is een workaround, maar dit stelt mogelijk wel je browser open voor aanvallen. Houd daar rekening mee! Je leest er hier meer over: http://chrome-allow-file-access-from-file.com/windows.html

## 7.5 Samenvatting

Het toevoegen van audio en video fragmenten is wat wij nu hebben geleerd. Er zitten haken en ogen aan wat betreft de browser ondersteuning en het gebruik van verschillende sources is een essentiele deal breaker. Ook het bieden alternatieve teksten of images bij niet kunnen laden van een fragment zal de finishing touch zijn voor de ervaring van de eindgebruiker.

# 8 Canvas en SVG

## 8.1 Inleiding

Het gebruik van de <canvas> en <svg> tags neemt geleidelijk aan toe. SVG, ook wel: Scalable Vector Graphics, worden in toenemende mate toegepast op webpagina's om op een visuele manier zaken te verduidelijken. Er zijn zelfs hele JavaScript bibliotheken geschreven die zich daarop richten. Waarop? Nou, denk bijvoorbeeld aan het maken van consistente, dynamische kolom grafieken, taart-grafieken, draaitabellen, UML diagrammen, enzovoorts.

Canvas wordt ook een steeds interessantere mogelijkheid om bijvoorbeeld HTML 5 games vorm te geven.

- Een canvas kunnen maken en daar een figuur op kunnen plaatsen met HTML
- Een canvas kunnen maken en daar een figuur op kunnen plaatsen met HTML

## 8.2 Het canvas element

Het canvas element, met bijbehorende <canvas> tag biedt ons de mogelijkheid te tekenen op een digitaal schildersdoek, in onze browser. Dit tekenen doen wij als ontwerpers van websites. Zoals je in bovenstaande inspiratiesessie hebt gezien, kan gebruik van het canvas element érg mooie resultaten geven.

Voor heel complexe tekeningen, is natuurlijk ook een gezonde hoeveelheid tekenvaardigheid nodig. In dit hoofdstuk en de daarbij behorende voorbeelden, zullen wij beginnen met het maken van onze eerste eenvoudige tekeningen.

Onderstaand voorbeeld beschrijft de verschillende mogelijke vormen, tekst en afbeeldingen die wij kunnen verwerken op een pagina. Je zult gaan zien dat er andere programmeercode nodig is naast HTML. Uiteraard is het wederom niet noodzakelijk de code tussen de <script> tags te kennen, maar wil je dit element echt kunnen doorgronden, is het minstens raadzaam de dikgedrukte teksten in onderstaand voorbeeld te proberen te begrijpen.

```
<html>
<head>
    <script>
      window.onload = function() {

document.getElementById("tekstknop").addEventListener(
"click",tekstdraw,false);

document.getElementById("lijnknop").addEventListener(
"click",lijndraw,false);

document.getElementById("cirkelknop").addEventListener(
"click",cirkeldraw,false);

document.getElementById("imageknop").addEventListener(
"click",imagedraw,false);
```

```
document.getElementById("kleurknop").addEventListener(
"click",kleurdraw,false);
        canvas = document.getElementById("canvas");
        canvas.style.border = "solid 1px black";
        tekengebied = canvas.getContext("2d");
        tmpImage = document.createElement("img");
        tmpImage.src = "https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/dend.jpg";
        tmpImage.width = "200";
        tmpImage.height = "100";
        tmpImage.style.display = "none";
        tmpImage.setAttribute("id", "plaatje");
      }

      function tekstdraw()
      {
        // Schrijven van tekst:
        tekengebied.font = "20px Tahoma";

        // "tekst", x-coordinaat, y-coordinaat
        tekengebied.fillText("Welkom bij Vijfhart",350,50);
        tekengebied.font = "italic 60px Tahoma";

        // "tekst", x-coordinaat, y-coordinaat
        tekengebied.fillText("!",520,60);
      }

      function imagedraw()
      {
        // Tonen van een afbeelding:
        document.body.appendChild(tmpImage);

        var plaatje = document.getElementById("plaatje");

        // plaatje, x-coordinaat, y-coordinaat
        tekengebied.drawImage(plaatje,210,200);
      }

      function lijndraw()
      {
        // Tekenen van een lijn:

        // definieren start (y,x)
        tekengebied.moveTo(400,200);

        // definieren eind (y,x)
        tekengebied.lineTo(200,100);

        // tekenen
        tekengebied.stroke();
      }
```

```
      function cirkeldraw()
      {
        // Tekenen van een cirkel:

        // defineren beginnen met vector tekening
        tekengebied.beginPath();

        // definieren boog (middenpunt x, middenpunt y,
radius
        // , beginpunt in graden op de cirkel, eindpunt in
graden op de cirkel)
        tekengebied.arc(168,75,40,0,2*Math.PI);

        // tekenen
        tekengebied.stroke();
      }

      function kleurdraw()
      {
        // Maken van een kleurverloop:
        // Create gradient
        var kleuren =
tekengebied.createLinearGradient(0,0,200,0);
        kleuren.addColorStop(0,"yellow");
        kleuren.addColorStop(1,"green");

        // Fill with gradient
        tekengebied.fillStyle = kleuren;

        // definieren start (x,y) en breedte en hoogte van
het vlak
        tekengebied.fillRect(10,150,200,150);
      }
    </script>
  </head>

  <body>
     <canvas id="canvas" width="600"
height="500"></canvas><br/><br/>
     <button id="tekstknop">Tekst toevoegen</button>
     <button id="imageknop">Plaatje toevoegen</button>
     <button id="lijnknop">Lijn toevoegen</button>
     <button id="cirkelknop">Cirkel toevoegen</button>
     <button id="kleurknop">Kleuren toevoegen</button>
  </body>
</html>
```

Zoals je in het bovenstaande voorbeeld hebt kunnen zien, het het canvas niet te 'besturen' met HTML code. Aanmaken ervan gaat goed, maar het dynamisch maken van een webpagina of onderdeel daarvan kan niet aan HTML overgelaten worden.

Om bovenstaande code grondig te begrijpen zal enige JavaScript kennis nodig zijn. Wij raden daarvoor je ook aan een kijkje te nemen op onze website, óf op de sit van MDN.

ebruik het bovenstaande voorbeeld en pas van alle items waarden aan. Let op: doe dit niet voor alle items in één keer. Begin bij de tekst en kijk wat er gebeurt als je daar een waarde tussen de dubbele quotes aanpast, of één van de nummers vervangt door een andere waarde.

Er staat ook commentaar in de code (regels die beginnen met: //), dit zijn tips die de gebruikte JavaScript code verduidelijken. Doe daarmee je voordeel.
Deze oefening is extra, mocht je er niet uitkomen, neem contact op met de docent.

Om een afbeelding toe te kunnen voegen aan een canvas, dient er reeds een afbeelding aanwezig te zijn op de webpagina. Let erop dat deze afbeelding geladen dient te zijn, alvorens je hem wilt gebruiken in het canvas. Het kán anders zo zijn, dat je de afbeelding nog niet ziet in je canvas, maar je ook geen foutmelding in de foutconsole ziet verschijnen.

Houdt er daarnaast rekening mee, dat gezien bovenstaande, het (vaak) logisch is de bestaande afbeelding op de pagina (die gebruikt wordt als bron voor het canvas), te verbergen. Dit kan middels CSS code:
display: "none";
óf met de code:
visibility: "hidden";
De eerste van deze opties zorgt ervoor dat het element ook daadwerkelijk geen ruimte op de pagina inneemt. Bij de tweede optie wel en kan de pagina er daardoor misvormd uit gaan zien.

## 8.3 SVG afbeeldingen

Scalable Vector Graphics, ook wel SVG genoemd, is een bestandsformaat waarmee afbeeldingen kunnen worden weergegeven. Deze afbeeldingen zijn opgebouwd uit een aantal coördinaten, kleuren en vullingen die vastgelegd zijn met behulp van XML tags (vergelijkbaar met HTML tags). Dit zorgt voor een interessant aantal voordelen voor deze afbeeldingen, want ze zijn:

- schaalbaar tot oneindige grootte zonder kwaliteitsverlies
- de afbeelding is 100% XML; een bekende standaard
- doordat de afbeelding uit XML code bestaat, is deze ook goed te scripten

Een SVG afbeelding is vaak niet té complex, omdat er anders te veel punten zijn die als coördinaten ingesteld dienen te worden om de afbeelding goed schaalbaar te maken, wat juist de kracht is van SVG. Denk aan grafieken!

Zelf maken van een SVG afbeelding is natuurlijk ook mogelijk. Gebruik hiervoor bij voorkeur een tool zoals Inkscape (open source) of Adobe Illustrator. Uiteraard zijn SVG afbeeldingen ook geheel in een tekst-editor te maken, maar dat is bijna niet te realiseren.

Een SVG afbeelding kan op meerdere manieren binnen een HTML pagina geïntegreerd worden, elk met haar eigen voordelen:
embedded als <svg> tag
verwijzing via een <img> tag
includen via de <embed> of <object> tag
Het is naast bovenstaande opties zelfs mogelijk de verwijzing naar een svg image als URL mee te geven aan een iframe. We zullen de bovenstaande voorbeelden hierna uitwerken. Let erop dat het <object> tag (zoals eerder benoemd) afgeraden wordt om te gebruiken.

Embedding via de <svg> tag is het meest veelzijdig indien wij de afbeelding nog willen aanpassen. Als wij de afbeelding ook op andere pagina's willen gebruiken óf de tag op een bepaald moment een andere bron willen geven, dan raden wij aan het <img> element of het <embed> element te gebruiken.

```
<html>
<head>
   <title>svg voorbeeld</title>
  </head>
<body>
        <svg width="300px" height="300px"
xmlns="http://www.w3.org/2000/svg">
            <g>
            <polygon points="40.586,20.586 32,29.172
23.414,20.586 20.586,23.414 29.172,
32 20.586,40.586 23.414,43.414 32,34.828 40.586,43.414
43.414,40.586 34.828,32 43.414,23.414    "/>
                <path
d="M32,4C16.537,4,4,16.537,4,32s12.537,28,28,28c15.463,
0,28-12.537,28-28S47.463,4,32,4z
M32,58C17.664,58,6,46.336,6,32

S17.664,6,32,6c14.336,0,26,11.664,26,26S46.336,58,32,58z"/>
            </g>
        </svg>

            <div id="andere_svg_methoden">
            <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/car.svg" />

                <embed type="image/svg+xml"
src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/car.svg" />
                <object type="image/svg+xml"
data="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/car.svg">
                Uw browser ondersteund geen SVG bestanden
            </object>
        </div>
    </body>
</html>
```

Onderstaand volgt nog een voorbeeld, waarbij wij middels SVG een staafdiagram hebben gemaakt. Ook hierin schuilt de kracht van SVG:

```
<html>
<head>
    <title>Voorbeeld van een SVG grafiek</title>
    <style>
      g > rect {
        fill: blue;
      }
    </style>
  </head>
  <body>
    <h2>Epicness</h2>
    <svg width="400" height="200">

      <title>epicness</title>

      <g>
        <rect width="50" height="18"></rect>
        <text x="55" y="8" dy=".35em">font element</text>
        <rect width="100" height="18" y="20"></rect>
        <text x="105" y="28" dy=".35em">span element</text>
        <rect width="75" height="18" y="40"></rect>
        <text x="80" y="48" dy=".35em">b element</text>
        <rect width="175" height="18" y="60"></rect>
        <text x="180" y="68" dy=".35em">em element</text>
      </g>
    </svg>
  </body>
</html>
```

Middels CSS of JavaScript code is het goed mogelijk om onze SVG bestanden anders op te maken zodra ze zijn ingeladen op onze webpagina. Denk hierbij aan het wijzigen van een aantal coördinaten of bijvoorbeeld het aanpassen van de achtergrondkleur van een driehoek.

Vanwege de 'scriptbaarheid' van SVG, wordt dit vaak gebruikt voor het maken van grafieken. Denk eens aan: gegevens uit een database worden via een web service beschikbaar gesteld en kunnen wij middels JavaScript uitlezen en in een dynamisch gegenereerde SVG grafiek plaatsen (denk eraan: SVG bestaat uit elementen en attributen die gewoon met JavaScript aan te passen zijn!).

## 8.4 Samenvatting

Met behulp van XML code die wij kunnen programmeren is een SVG file ontstaan. Ook kunnen wij zelf figuren laten tekenen in een HTML canvas. We weten wat de verschillen zijn tussen beide technieken en wanneer wij het beste welke techniek kunnen toepassen.

# 9 HTML5 API's

## 9.1 Inleiding

De onderwerpen die in dit laatste hoofdstuk van deze training aan bod komen, vallen qua kennisgebied in HTML, ook binnen JavaScript. Dit betekent dat wij in een boven gemiddeld aantal oefeningen JavaScript code gaan gebruiken om deze nieuwe HTML 5 features van dit hoofdstuk te kunnen toepassen.

Wij gaan kijken naar verschillende API's, ofwel: Application Programming Interface's. Dat is een algemene term, bedoelt om te verwijzen naar stukken programmatuur die gemaakt is om functionaliteit toe te voegen aan een systeem.

Met de komst van HTML 5 zijn er vele API's bijgekomen. Het is ook goed om te beseffen dan HTML nog steeds in ontwikkeling is, dus er ook continu API's bijgeschreven worden.

- Het maken van web applicaties die het gebruik van offline functionaliteit mogelijk maakt.
- Web sockets gebruiken om data van en naar een web applicatie en web server te sturen.
- De usability van een web applicatie verbeteteren door langlopende processen te laten behandelen door web workers.

## 9.2 Ofline applicaties

Om te voorzien in de groeiende behoefte aan grotere en veelzijdigere online applicaties, zijn er met HTML 5 een aantal erg praktische performance features bij gekomen:

- application cache
- localStorage
- web SQL & indexed database

Wij gaan in dit hoofdstuk  in ieder geval ieder van deze aspecten behandelen.
Application cache

Het cachen van bestanden wordt binnen de IT wereld gebruikt om performance van applicaties en systemen te verbeteren. Een grote afbeelding op een webpagina kan bijvoorbeeld gecached worden, ofwel: opgeslagen wordt op de harde schijf van de eindgebruiker. Hierdoor hoeft dit bestand bij een volgend bezoek niet meer gedownload te worden.

Er zijn grofweg drie gegronde redenen om gebruik te maken van offline storage met behulp van de application cache:

- Performance: files laden sneller van disk dan van Internet
- Backup: bij uitval van files zijn er offline bestanden beschikbaar
- Offline browsing: site blijft bruikbaar bij verlies Internetconnectie van de eindgebruiker

De browsercache is tegenwoordig in de meeste browsers in zogeheten containers opgeslagen. Dat zijn bestanden, voor de eindgebruiker niet bruikbaar, waarin de te cachen files worden opgeslagen. Onherkenbaar aan de buitenkant qua filename of filesize.

Waar wordt een gedownload bestand opgeslagen? Dat verschilt per besturingssysteem én per browser. Één van de meest generieke manieren om de inhoud van de browser cache te zien, is om in de browser URL / adres balk in te vullen: about:appcache-internals. We krijgen dan de inhoud van de bovengenoemde containers te zien.
Met about:about zijn alle mogelijkheden te zien.

Willen wij écht zelf naar de containers of files op zoek gaan, kijk dan in de volgende directories per browser:

- In Windows 7: C:\Users\username\AppData\Local\Microsoft\Windows\Temporary Internet Files
- In Windows 8: C:\Users\username\AppData\Local\Microsoft\Windows\INetCache

- In Windows 7: C:\Users\<username>\AppData\Local\Mozilla\Firefox\Profiles\<salt>.<profile name>\OfflineCache
- In Linux en Mac OS: /Users/<username>/Library/Caches/Firefox/Profiles/<salt>.<profile name>/OfflineCache

- In Windows 7: C:\Users\[USERNAME]\AppData\Local\Google\Chrome\User Data\Default\Cache
- In Linux en Mac OS: /Users/[USERNAME]/Library/Caches/Google/Chrome/

- In Linux en Mac OS: /Users/<gebruiker>/Library/Caches/com.apple.Safari/WebKitCache/Version 4/Blobs /Users/<gebruiker>/Library/Caches/com.apple.Safari/WebKitCache/Version 4/Records /Users/<gebruiker>/com.apple.Safari/Cache

- In Windows 7: C:\Users\Administrator.LP-DOCENT-SJ\AppData\Local\Opera Software\Opera Stable
- In Linux / Mac OS: /home/<gebruiker>/.opera
-

Willen wij écht zelf naar de containers of files op zoek gaan, kijk dan in de volgende directories per browser:

In Windows 7: C:\Users\username\AppData\Local\Microsoft\Windows\Temporary Internet Files
In Windows 8: C:\Users\username\AppData\Local\Microsoft\Windows\INetCache

In Windows 7: C:\Users\<username>\AppData\Local\Mozilla\Firefox\Profiles\<salt>.<profile name>\OfflineCache
In Linux en Mac OS: /Users/<username>/Library/Caches/Firefox/Profiles/<salt>.<profile name>/OfflineCache

In Windows 7: C:\Users\[USERNAME]\AppData\Local\Google\Chrome\User Data\Default\Cache
In Linux en Mac OS: /Users/[USERNAME]/Library/Caches/Google/Chrome/

In Linux en Mac OS: /Users/<gebruiker>/Library/Caches/com.apple.Safari/WebKitCache/Version 4/Blobs

/Users/<gebruiker>/Library/Caches/com.apple.Safari/WebKitCache/Version 4/Records
/Users/<gebruiker>/com.apple.Safari/Cache

In Windows 7: C:\Users\Administrator.LP-DOCENT-SJ\AppData\Local\Opera Software\Opera Stable
In Linux / Mac OS: /home/<gebruiker>/.opera
Hoe weet de browser welke bestanden hij lokaal op de machine van de eindgebruiker dient op te slaan? Dat is afhankelijk van de waarden die in een zogeheten manifest file staan benoemd. Daarin is aan te geven welke files gecached dienen te worden (cache categorie) , maar ook welke files in geval van verlies van Internet verbinding als backup (fallback) gebruikt worden. Tot slot kunnen wij ook aangeven welke files er in ieder geval niet via de cache mogen lopen (indien er verbinding is met een website) , deze vallen onder de categorie network.

Een manifest file begint altijd met de tekst CACHE MANIFEST. Dit dient gevolgd te worden door een <enter> of simpelweg een nieuwe regel. Vervolgens kunnen de files benoemd worden die gecached dienen te worden (één filenaam per regel). Hierbij mag zowel een relatief pad gebruikt worden, als een absoluut pad naar een andere website (middels: http:// notatie). Alle benoemde files vallen standaard onder de cache categorie.

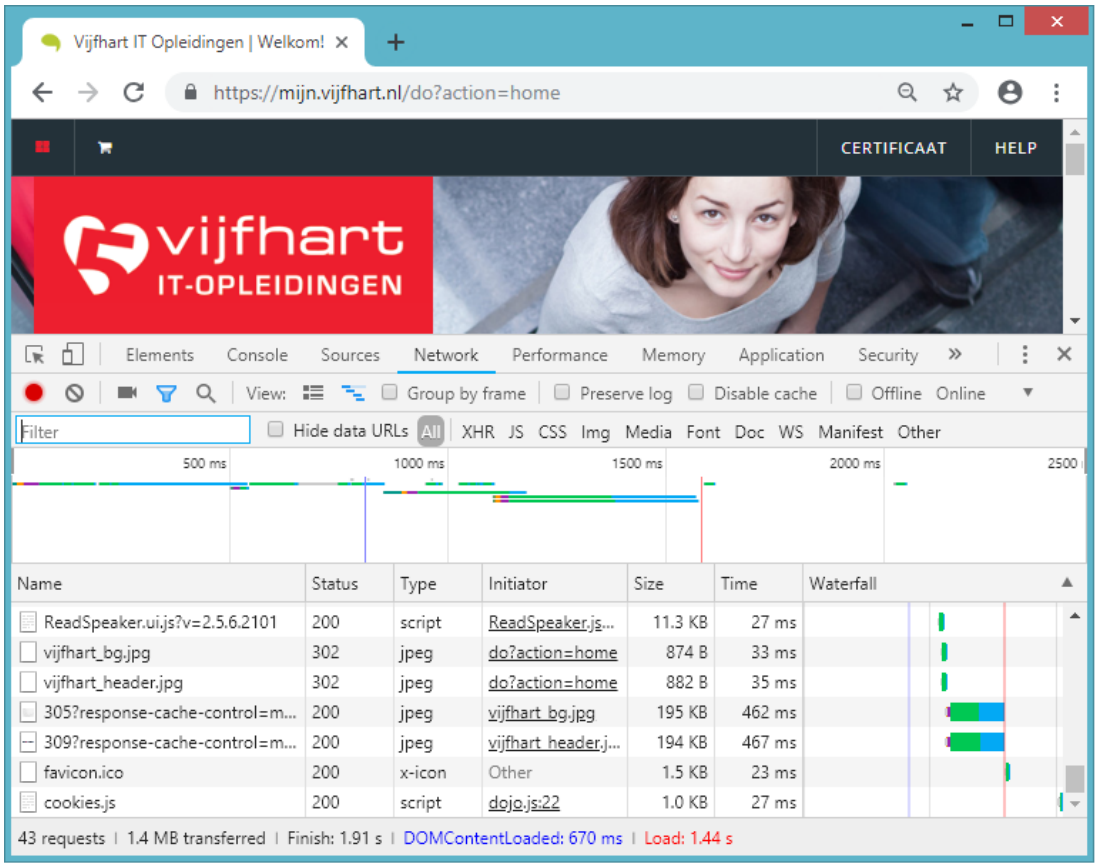
```
CACHE MANIFEST
# Gemaakt door Vijfhart-IT
# Commentaar regels beginnen met een hash-teken
index.html
producten.html
images/groot_plaatje.jpg
scripts/jquery.js
```

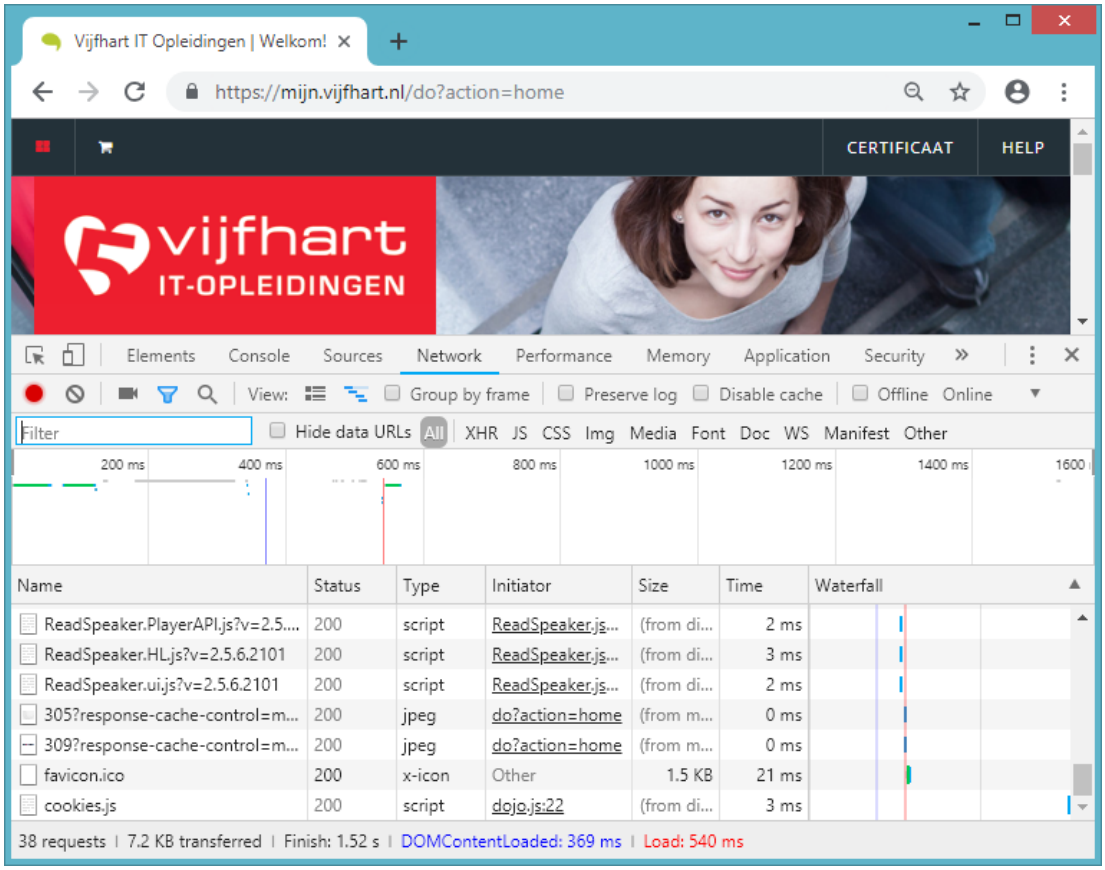
```
CACHE MANIFEST
# Gemaakt door Vijfhart-IT
# Commentaar regels beginnen met een hash-teken
CACHE:
index.html
producten.html
images/groot_plaatje.jpg
scripts/jquery.js

NETWORK:
database_verzoeken.php
login_check.php

FALLBACK:
index.html
belangrijk.html
```

Begrijp je hoe bovenstaande voorbeelden werken? Probeer dan zelf ook een manifest bestand aan te maken. Om het effect ervan te kunnen testen dien je online een webpagina te hebben gehost of lokaal een echte webserver te draaien(!) Hoe kan je het testen? Ga naar de 'developer tools' in je browser (middels de F12 knop) en kijk of de bestanden die je wilt cachen volgens je manifest file, bij je downloads staan op het tabblad 'network'.

# 9 HTML5 API's

Bekijk onderstaande screenshots van de Vijfhart website. Bij het eerste screenshot is er nog niets gecached. Bij het tweede screenshot wel. Onderaan de screenshots ziet u (in rood) hoeveel tijd het laden van losse files van deze pagina heeft gekost.






Zie de balk(en) onderin de screenshots: een verschil van 900 ms! Dat klinkt misschien als weinig, maarbesef dat dit een pagina is die (onder andere) weinig images hoeft te laden. En toch is de pagina al bijna 2/3e deel sneller dan wanneer files niet gecached worden.

Is één van je te cachen bestanden gewijzigd en willen wij dat deze opnieuw gedownload wordt door de eindgebruikers? Wijzig dan je manifest bestand door er (bijvoorbeeld) een versienummer in commentaar (#) in te zetten. Doordat de manifest file dan gewijzigd is, wordt deze door de browser opnieuw doorlopen en verwerkt. En worden daarmee eventueel gewijzigde cache files opnieuw gedownload. Anders gebeurt dit niet.

Om goed met een manifest bestand te kunnen werken, dient ook uw webserver goed ingesteld te zijn. Voor meer informatie over manifest files én het juist configureren van server instellingen, bekijk de MDN website.

### 9.2.1 Local storage API

Met local storage, bedoelen wij het lokaal op de eindgebruiker zijn device opslaan van gegevens voor later gebruik. Denk aan de ouderwetse cookies waar je ongetwijfeld al eens van hebt gehoord. Deze waren echter vrij onveilig. Dat is met local storage minder het geval. Middels local storage kunnen wij zogeheten key-value paren opslaan. Denk bijvoorbeeld aan:

- BSN nummer - Naam
- Barcode - Titel

- Voornaam - Geboortedatum
- Kantoornummer - Kantoornaam

Wanneer wij local storage willen gaan gebruiken, moeten wij kiezen uit een tweetal implementaties:

- sessionStorage, dat is lokale opslag, die verloren gaat zodra de eindgebruiker de browser afsluit.
  Een voordeel van deze manier van opslag is dat er geen (eventueel) gevoelige data op de computer van de eindgebruiker blijft opgeslagen nadat de webbrowser is afgesloten.
- localStorage, dat is eveneens lokale opslag, die behouden blijft wanneer de browser wordt afgesloten en/of heropend.
  Een voordeel het localStorage mechanisme, is dat we bijvoorbeeld gebruikersprofielen (instellingen, winkelwagentjes etcetera) kunnen behouden op de computer, waardoor deze zelfs nog beschikbaar zijn als de webbrowser afgesloten is geweest en de gebruiker bijvoorbeeld na 2 weken weer verder wilt shoppen met hetzelfde winkelwagentje.

Via de gelijknamige eigenschappen (of eigenlijk objecten) window.sessionStorage en window.localStorage kunnen wij gebruik maken van deze opslag methoden. Voor beide methoden zijn er een tweetal functies geschreven die wij op het sessionStorage en localStorage object kunnen aanroepen. Meer over de werking van objecten, eigenschappen en functies vindt je in onze JavaScript cursus.

In onderstaand voorbeeld gaan wij de twee functies (setItem en getItem) toepassen. Beide functies accepteren parameters. Dit zijn in het geval van de setItem() functie twee parameters: de key en de value. In het geval van de getItem() functie (dus om een waarde uit de storage te halen), hebben wij alleen een key nodig. Daarmee wordt voor ons de bijbehorende value opgehaald.

```
<html>
<head>
    <title>HTML 5 local storage</title>
    <script>
    window.onload=function(){

    session = sessionStorage;
    local = localStorage;

    // Controle of sessionStorage al gevuld is bij 'naam' entry
    if (session.getItem("naam"))
    {
      alert("Uw met sessionStorage opgeslagen naam is: " +
session.getItem("naam"));
    }

    // Controle of localStorage al gevuld is bij 'naam' entry
    if (local.getItem("naam"))
    {
      alert("Uw met localStorage opgeslagen naam is: " +
local.getItem("naam"));
    }
    // Toevoegen actie button 1:

document.getElementById("btnSession").addEventListener("click",function(){
```

```
      var invoer = document.getElementById("txtSession");
      if (invoer != "")
      {
        // Instellen 'naam' entry met de waarde van het invoerveld
        session.setItem("naam", invoer.value);
      }
    },false);

    // Toevoegen actie button 2:

document.getElementById("btnLocal").addEventListener("click",function(){
      var invoer = document.getElementById("txtLocal");
      if (invoer != "")
      {
        // Instellen 'naam' entry met de waarde van het invoerveld
        local.setItem("naam", invoer.value);
      }

    },false);

    }
    </script>
  </head>
  <body>
    <fieldset>
      <caption>sessionStorage</caption>
      Voer uw naam in:<br/>
      <input type="text" id="txtSession" />
      <button id="btnSession">Opslaan</button>
    </fieldset>

    <fieldset>
      <caption>localStorage</caption>
      Voer uw naam in:<br/>
      <input type="text" id="txtLocal" />
      <button id="btnLocal">Opslaan</button>
    </fieldset>
  </body>
  </html>
```

Om dit te zien werken kun je in chrome de developer tools onder application kijken. Je kunt het ook zien werken door nadat je wat hebt ingevuld de pagina te verlaten en weer te openen. Dan zal de localStorage in een alert worden getoond.

Om met JavaScript af te vangen of er (door een bepaalde taak) een item is aangepast in één van beide storage mechanismen, kunnen wij gebruik maken van het window.onstorage event. Daar zullen wij dan een zogeheten event listener (let op: JavaScript!) voor moeten schrijven. Dit valt wegens de grote hoeveelheid aan JavaScript code, buiten deze cursus.

Mocht je dit toch interessant vinden, probeer dan het volgende voorbeeld:

```
<html>
<head>
  <title>storage voorbeeld</title>
</head>
<body>
    <table>
      <tr>
        <td>Key:</td>
        <td><input type="text" id="txtKey" /><td/>
      </tr>
      <tr>
        <td>Value:</td>
        <td><input type="text" id="txtValue" /><td/>
      </tr>
      <tr>
        <td colspan="2">
          <button id="btnAdd">Voeg toe</button>
        </td>
      </tr>
      <tr>
        <td colspan="2">
          <button id="btnDelete">Wis lokale storage</button>
        </td>
      </tr>
    </table>


    <table id="inhoud">
      <tr>
        <th colspan="3">Inhoud localStorage:</th>
      </tr>
      <tr>
        <th>Key</th>
        <th>Value</th>
      </tr>
    </table>
  <script>
  function refresh()
      {
        alert("refresh!");
        var tabel = document.getElementById("inhoud");
        var rijen = tabel.getElementsByTagName("tr");
        alert(rijen.length);

        for (i=0; i<rijen.length; i++)
        {
          if (i!=0 && i!=1)
          {
            tabel.removeChild(tabel.childNodes[i]);
          }
        }

        for (i=0; i<localStorage.length; i++)
```

```
        {
          var rij = document.createElement("tr");
          var kolom1 = document.createElement("td");
          var kolom2 = document.createElement("td");

          var keyEntry = localStorage.key(i);
          var valueEntry = localStorage.getItem(keyEntry);

          var resultaat1 = document.createTextNode(keyEntry);
          var resultaat2 = document.createTextNode(valueEntry);

          kolom1.appendChild(resultaat1);
          kolom2.appendChild(resultaat2);
          rij.appendChild(kolom1);
          rij.appendChild(kolom2);
          tabel.appendChild(rij);
        }
      }

      window.onload=function(){


document.getElementById("btnAdd").addEventListener("click",function(){

          var keyEntry = document.getElementById("txtKey").value;
          var valueEntry = document.getElementById("txtValue").value;
          if (keyEntry != "" && valueEntry != "")
          {
            localStorage.setItem(keyEntry,valueEntry);
            refresh();
          }
          else
          {
            alert("Graag een waarde invoeren voor de 'key' en de
'value'");
          }
        },false);


document.getElementById("btnDelete").addEventListener("click",function(){
          localStorage.clear();
          refresh();
        },false);
      }
  </script>
</body>
</html>
```

Ook hier kun je in Chrome met de developertools de waarden weer volgen onder application.

Benadering van de locale opslag (ongeacht het mechanisme) via een iframe vanuit een andere webpagina is niet toegestaan indien de 'disable 3rd party cookies' optie is ingeschakeld in de browser.

### 9.2.2 Webdb en indexed db

Deze mechanismen zijn bedoeld om te gebruiken wanneer wij grote hoeveelheden data lokaal op de machine van de eindgebruiker willen opslaan, gestructureerd willen gebruiken en eventueel willen doorzoeken.

WebDB wordt geadviseerd om niet meer te gebruiken, omdat dit niet generiek toepasbaar is. Omdat WebDB SQL gebruikt als database taal, zou iedere ondersteunende browser hetzelfde SQL dialect moeten gaan gebruiken. Dat is niet realiseerbaar. Daarnaast is het concept van een relationele database met tabellen niet meer praktisch in een HTML / JavaScript omgeving die vlot moet werken.

IndexedDB slaat al zijn gegevens gewoon op als objecten, waar prima met Javascript mee te werken is, zij het iets complexer. Het maakt gebruik van een zogeheten NoSQL database. Omdat beide concepten sterk rusten op het gebruik van JavaScript, verwijzen wij hiervoor enkel naar de MDN webpagina voor meer informatie hierover: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API .

Het bekijken van de inhoud van  WebDB en Indexed DB, kan net als voor de local storage opties, in de developer tools Google Chrome onder het tabblad 'application'.

## 9.3 Usability API's

### 9.3.1 Page visibility API

Deze API stelt ons in de mogelijkheid te controleren of de eindgebruiker onze webpagina momenteel in beeld heeft en er geen ander tabblad in beeld is momenteel.

De page API werkt met behulp van een tweetal eigenschappen van het document object genaamd 'hidden' en 'visibilityState'. De eerste kan als waarde 'true' of 'false' hebben en daarmee weten wij als programmeur dus of de pagina momenteel zichtbaar is voor de eindgebruiker. Het attribuut visibilityState kan al waarde hebben: 'hidden', 'visible', 'prerender' of 'unloaded'.

Om in de praktijk goed te kunnen werken met deze API, is er een zogeheten event gekoppeld aan de verandering van de visibilityState. Dat is een signaal dat wordt gegeven wanneer (bijvoorbeeld) de gebruiker op een ander browser tabblad klikt. Dit signaal is met JavaScript code op te pakken zodat wij erop kunnen reageren. Een popup die verschijnt zodra wij naar een andere pagina willen navigeren of indien er een HTML ruimtevaart spel open staat en continu in beeld wilt blijven ("Please don't go, the drones need you").

Wanneer je zware taken op je webpagina hebt lopen die essentieel zijn indien de gebruiker op je pagina is. Maar zodra je gebruiker even op een andere webpagina is, of even in een andere applicatie of game iets gaat doen, hoeft je zware taak wellicht minder te doen. Je kan daarmee de performance van je pagina verbeteren en daarmee de ervaring voor je eindgebruiker.

n onderstaand voorbeeld demonstreren wij het effect van de visibility API. Ook hier is weer de nodige JavaScript kennis nodig. We maken eerst een functie die afhankelijk van de staat van de pagina (zichtbaar of onzichtbaar), deze gelijknamige staat laat zien in de titel van de pagina (naam van het tabblad!).

Het tweede deel van de (JavaScript) code zorgt ervoor dat het visibility event goed wordt afgevangen in alle gangbare browsers.

```
<HTML>
  <head>
    <title>HTML 5 page visibility</title>
    <script>

    /* Werkt niet goed in CodePen */

    window.onload = function() {
      function handleVisibilityChange() {
        if (document[hidden]) {
        document.title = "onzichtbaar";
        } else {
        document.title = "zichtbaar";
        }
      }
      // Instellen van de eigenschap, zodat dit in de
meeste browsers werkt:
      var hidden, visibilityChange;
      if (typeof document.hidden !== "undefined") { //
FireFox en Opera
        hidden = "hidden";
        visibilityChange = "visibilitychange";
      } else if (typeof document.msHidden !== "undefined")
{ // Internet Explorer
        hidden = "msHidden";
        visibilityChange = "msvisibilitychange";
      } else if (typeof document.webkitHidden !==
"undefined") { // Google Chrome
        hidden = "webkitHidden";
        visibilityChange = "webkitvisibilitychange";
      }

      // Toevoegen van de zogeheten event listener
      document.addEventListener(visibilityChange,
handleVisibilityChange, false);
    }
    </script>
  </head>
      Open meer tabbladen en kijk naar de titel op de tab
  <body>

  </body>
</HTML>
```

### 9.3.2 Fullscreen API

Hiermee kunnen wij aangeven dat wij een bepaald deel van onze pagina op het volledige scherm willen tonen (bijvoorbeeld na het klikken op een link of ingeven van een bepaalde toets). Hiertoe gebruiken wij middels JavaScript de requestFullscreen() functie van een element. Er wordt dan aan de gebruiker gevraagd of hij / zij wilt overschakelen naar een

volledig-scherm weergave. Het afsluiten van een volledig scherm kan de gebruiker doen met <ESC> of <F11>. Ook kan dit programmeertechnisch door de exitFullscreen() functie aan te roepen, bijvoorbeeld na het klikken op een button met een kruisje.

Deze API functies (verzoek tot openen en het daadwerkelijk afsluiten van fullscreen mode) dienen altijd aangeroepen te worden vanuit een JavaScript event handler. Dit zorgt ervoor dat de code netjes blijft en men minder snel kwaadaardige streken uit kan halen.

Let op dat er in browsers verschil zit in de aanroep van de functie om een object fullscreen te tonen. We kunnen dat in de volgende browsers doen met:

- Chrome / Opera: webkitRequestFullscreen()
- FireFox: mozRequestFullScreen()
- Internet Explorer: msRequestFullscreen()
- Edge / Safari: webkitRequestFullscreen()

Voor meer informatie over browser specifieke verschillen zie: MDN. In het volgende voorbeeld gaan wij voor het (JavaScript) gemak uit van de Google Chrome browser.

```
<head>
    <title>HTML 5 fullscreen video voorbeeld</title>
    <script>
      window.onload = function(){

document.getElementById("full").addEventListener("click",function(){

document.getElementById("filmpje").webkitRequestFullscreen();
        },false);
      }
    </script>
  </head>
  <body>
    <video controls>
      <source src="5hart_muurschildering.mp4" type="video/mp4"
id="filmpje">

      Het afspelen van deze video wordt door uw browser niet
ondersteund.
      <img src="http://cdn3.wpbeginner.com/wp-
content/uploads/2015/08/error.jpg">
    </video>

    <br/>
    <button id="full">Fullscreen<button>
  </body>
```

### 9.3.3 Geolocation API

Ook dit is een erg toepasbare API. Hiermee kunnen wij de geografische coordinaten van het device van de eindgebruiker opvragen. Indien de eindgebruiker akkoord geeft, krijgen wij deze informatie door en kunnen wij hiermee onze applicatie aansturen.

Voorbeeld toepassingen:

- Verlenen van accurate file informatie
- Mapping van winkelgebieden

- Jacht naar Pokémon
- Zoeken van geocaches (schatzoeken)

En zo zijn er nog tal van voorbeelden te bedenken. Voor deze api geldt, helaas, dat ook hierin voornamelijk JavaScript code gebruikt wordt. Om dus gebruik te kunnen van deze API raden wij sterk aan de nodige JavaScript kennis op te doen en wat betreft deze API meer informatie hierover te lezen op MDN.

### 9.3.4 File API (Drag & Drop)

Deze uitgebreide API bied ons de mogelijkheid bestanden die een eindgebruiker geselecteerd, uit te lezen. Enkele voor de hand liggende toepassingen zijn:

- laten selecteren van bestanden om te uploaden naar een online foto album
- opsturen van een pdf bestand met daarin uw identiteitsbewijs voor de boeking van een hotel overnachting
- kiezen van een Excel bestand of komma gescheiden bestand om in te laden in een online database
- Selecteren van tekst om in een veld te plakken (knippen-plakken)

Het daadwerkelijk laten selecteren van bestanden kan op twee manieren:

- Met een input element, met als een type="file" attribuut, dus: <input type="file" />
- Ook kan het middels 'drag and drop'. Dit is een mechanisme wat erg in opkomst is geraakt sinds de mogelijkheid om met JavaScript het zogeheten Document Object Model (DOM) te manipuleren.

Drag and drop is niet alleen van toepassing op bestanden, maar inhoud van HTML elementen zoals tekst en afbeeldingen kunnen ook prima hiervoor worden gebruikt.

We maken een element draggable (sleepbaar) door er een attribuut draggable aan toe te voegen, met als waarde "true". Om te monitoren of er iets 'opgepakt' en/of ergens 'gedropt' wordt, zijn er een aantal events die wij kunnen afvangen middels JavaScript. Wij zullen met JavaScript code moeten luisteren naar het 'dragstart' en het 'drop' event. Het te slepen object is binnen JavaScript gerepresenteerd door het dataTransfer object. Deze heeft een setData() en een getData() methode om respectievelijk gegevens (tekst, files, images) eraan toe te voegen en om data uit het object in te lezen (bijvoorbeeld om het te plaatsen in het element waar wij onze tekst willen 'droppen').

Bij het drag & droppen van een bestand naar een browser, zorg er dan voor dat in het event dragover de methoden stopPropagation() én preventDefault() gedraaid zijn op het body element! Doen we dat niet, dan zal de browser als default gedrag de file(s) proberen te openen.

## 9.4 Mobiele device API's

### 9.4.1 Navigator object

Binnen een browser zijn allerlei objecten gecreerd die wij middels JavaScript code kunnen benaderen. Zoals een window object en een navigator object. Waarbij de laatst genoemde onder andere eigenschappen bevat over welk type browser de eindgebruiker mee werkt. Veel van de volgende API's zijn onderdeel van het window object of het navigator object. Om daar goed mee te kunnen werken, is enige JavaScript kennis aan te bevelen.

Wij zullen daarom ook gedurende het restant van dit hoofdstuk enkel wat voorbeelden beschrijven. Opdrachten maken is hierbij niet nodig.

### 9.4.2 Vibrate API

Een API die vooral binnen gaming zijn toepassing vindt is de vibrate API. De uitwerking van deze API kan ervoor zorgen dat het (mobiele) device waarmee de eindgebruiker de webpagina benadert, begint te trillen.

Hiertoe heeft het navigator object een functie genaamd vibrate() gekregen. Daaraan kunnen wij optioneel meegeven hoe lang het device dient te trillen in milliseconden (1000 milliseconden = 1 seconde).

```
<html>
<head>
    <script>
      window.onload=function(){

document.getElementById("startbtn").addEventListener("click",function(){
          window.navigator.vibrate(4000);
        },false);
      }
    </script>
  </head>
  <body>
    Klik <button id="startbtn">hier</button> om de hobbelige weg op te
rijden...
  </body>
  </html>
```

Indien je bovenstaand voorbeeld uitvoert op een mobiel device dat kan trillen, dan zal je het effect merken. Je kunt deze code bijvoorbeeld online brengen door het te kopiëren naar: Jsfiddle (https://jsfiddle.net/), het daar op te slaan en vervolgens met een mobiel device naar de betreffende URL navigeren. Leuk om te proberen!

### 9.4.3 getUserMedia API

Om gebruik te kunnen maken van media apparaten op het device van de eindgebruiker, kunnen wij de getUserMedia API gebruiken. Deze vraagt de gebruiker of wij gebruik mogen maken van zijn of haar camera en/of microfoon. Hierdoor kunnen wij bijvoorbeeld live filmen of foto's laten maken. In context: schade rapportage app, een videochat applicatie, of voor het opnemen van een virtuele speelomgeving (denk aan augmented reality).

### 9.4.4 Battery level API

Tot slot kennen we ook een battery level API. Deze kan de status van de batterij van het device van de eindgebruiker opvragen. Erg handig om een app bijvoorbeeld in een 'low energy' state te kunnen zetten. Helaas is de ondersteuning door de browsers enkel nog beperkt tot Firefox. Dat is ook de reden dat wij er hier niet meer aandacht aan zullen besteden.

## 9.5 Websockets

Met WebSockets is het mogelijk een communicatielijn open te zetten van een client browser naar een server. Via deze lijn kunnen we data heen en weer sturen. De afhandeling van het

versturen, ontvangen en de tussentijdse status daarvan is geregeld middels JavaScript events.

Het aanmaken van een WebSocket object gaan wij zo onderstaand tonen. Om dit te kunnen testen dient er een server beschikbaar te zijn die onze connectie accepteert. Dat kan een Node.js server zijn of bijvoorbeeld een PHP server. Ook dient de server een WebSocket omgeving te hebben draaien.

Let dus op: we kunnen niet zondemeer een WebSocket object maken en data proberen te versturen. Er zijn de nodige installatiestappen nodig op een server.

Onderstaand volgt de code voor het aanmaken van een WebSocket object:

```
//onveilige optie, die niet gebruik maakt van een zelf
bedacht hand-shake sleutelwoord:
var client = new WebSocket('ws://localhost:8080/');

//betere variant die gebruik maakt van een zelf bedacht
sleutelwoord om als verificatie te gebruiken:
var client = new WebSocket('ws://localhost:8080/',
'plaatjes_zenden');
```

Op http://jsfiddle.net/markodraisma/8fLbubax/ staat een voorbeeld op een server klaar.

## 9.6 Web Workers

Web workers zijn werktaken die wij kunnen starten, en parallel kunnen laten lopen aan de rest van de webpagina. Denk bijvoorbeeld aan grote berekeningen die er moeten gebeuren wanneer wij door een 3D heelal simulatie bladeren. Deze berekeningen kunnen het best op de achtergrond gebeuren, zodat de rest van de website er niet op hoeft te wachten. Wat wél interessant is, is om tussentijds informatie van de web worker te krijgen, over bijvoorbeeld hoeveel m3 de planeet Jupiter is na een herberekening.

Iedere webworker krijg op de processor van de computer zijn eigen 'thread', ofwel core toegewezen. Tenminste, dat is de bedoeling. In de praktijk heeft de processor nog weleens andere plannen. Dus dat is nooit geheel met zekerheid te zeggen.

Een webworker kan een willekeurig JavaScript bestand draaien. Deze filenaam dienen we op te geven bij het aanmaken van een webworker. Er kunnen meerdere webworkers gelijktijdig actief zijn en de kracht zit hem naast dat ze ogenschijnlijk parallel uitgevoerd worden, er ook berichten uitgewisseld kunnen worden tussen de 'starter' van de webworker en de worker zelf. Deze communicatie is evenals bij WebSockets te managen via events.

Enkele voorbeeldtoepassingen van webworkers:

- uitzoeken in een database van welke artiest een opgenomen audio sample is
- omzetten van een grote hoeveelheid afbeeldingen naar 'zwart-wit'
- berekenen van alle priemgetallen tot 10 miljoen, terwijl de pagina responsive blijft

Hoe maken wij een webworker aan? Onderstaand kunnen wij een voorbeeld downloaden. Wij hebben een tweetal bestanden aangemaakt. Een HTML file waarmee wij een webworker aanmaken. Deze webworker gebruikt een JavaScript bestand waarin alle priemgetallen t/m 100 worden berekend. Heeft hij een priemgetal gevonden, dan zal hij deze direct terugsturen naar de HTML pagina en tonen op het scherm.

```
<html>
  <head>
  <script src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/priem.js"></script>
</head>
<body>

        <button>start</button>

        <div id="resultaat">Priemgetallen t/m 1000:</div>
<script>
            window.onload=function(){

                rekenmachine = new Worker(URL.createObjectURL(new
Blob(["("+priempje.toString()+")()"], {type: 'text/javascript'})));

                var resultaat = document.getElementById("resultaat");

                rekenmachine.addEventListener("message",function(e){

                    console.log('priem getal ontvangen!');

                    resultaat.innerHTML = resultaat.innerHTML += "<br/>" +
e.data;

                }, false);



document.getElementsByTagName("button")[0].addEventListener("click",function(){

                    rekenmachine.postMessage(1000);

                },false);

            }</script>
</body>
</html>
```

Om te zien wat er in het JavaScript bestand 'priem.js' staat, kun je de URL kopieren zoals deze in het 'src' attribuut van het element <script> staat.

Omdat WebWorkers met meerdere threads kunnen werken, hebben ze niet de beschikking over alle mogelijke JavaScript features, maar wél over:

- navigator object
- location object
- XMLHttpRequest (server verzoeken)
- Timeout() methoden
- Application cache
- Scripts importeren met een importScripts() methode

Webworkers kunnen en mogen niet bij:

- het Document Object Model (opbouw van de pagina, omdat deze niet veilig bijgewerkt kan worden in geval van meerdere threads gebruik gemaakt wordt)
- window object
- document object
- een parent object

Meer informatie over het werken met web workers leest u ook op MDN. Een leuke oefening ermee lees je op w3schools.

### 9.7 Samenvatting

Wij hebben gemerkt dat er een groot aantal API's zijn die het gebruik van HTML de moderne wereld in trekken. API's gericht op performance, op mobiele devices en natuurlijk vooral op de optimale gebruiker beleving. Ongetwijfeld is opgevallen dat hierbij veel JavaScript code komt kijken en dat de grens tussen enkel HTML programmeren en het gebruik van JavaScript fijn is.