IK WIL

Webpagina's ontwikkelen met HTML 5

vijfhart IT-OPLEIDINGEN

Inspiratie sessie

https://webflow.com/blog/20-web-design-trends-for-2019

Introductie

1. Voorstelrondje
2. Agenda
3. Dagindeling
4. Cursusmateriaal
5. Doel van deze cursus

Agenda dag 1 en 2

Dag 1

- H1 Inleiding
- H2 Elementen en attributen
- H3 Opmaak
- H4 Afbeeldingen, hyperlinks en embedding

Dag 2

- H5 Tabellen, lijsten en speciale karakters
- H6 Formulieren
- H7 Audio en video
- H8 Canvas en SVG

Dagindeling

- 8:45 - ..... Begin cursusdag
  - Introductie
  - H1 Algemeen
  - H2 Hoofdstukinhoud
    - Theorie
    - opdrachten maken
    - opdrachten bespreken
- 10:30 - 10:45 Koffiepauze
- 12:00 - 12:45 Lunchpauze
- 14:30 - 14:45 Koffiepauze
- ..... - 16:00 Einde cursusdag

Cursusmateriaal

- Online leeromgeving
  - PDF met sheets
  - Content uitgeschreven
- CodePen.io oefen omgeving
- Bonus materiaal

Doel van deze cursus

- 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.

H1 - Inleiding

H1 - Inleiding

1 Doelstelling

2 Handige websites

3 Tools

4 Introductie HTML

5 Introductie W3C

Doelstelling

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).

Wat zijn JavaScript en CSS?

Handige websites

Een kleine greep uit een selectie met praktische websites om tijdens en na deze cursus bij de hand te houden:

- HTML en JavaScript naslag: MDN
- Browser ondersteuning: Can I use it?
- Browser ondersteuning HTML 5: HTML 5 test
- Browser ondersteuning CSS en Javascript: Quirksmode
- Web development oefeningen: W3 schools
- Web development tips: HTML 5 rocks
- Innovatieve HTML 5 websites: FWA awards

Tools

Tijdens deze training gebruiken wij:

- Google Chrome, met Developer Tools

Qua editor(s) kunnen we gebruik maken van:

- NotePad++ (basic)
- Visual Studio Code (gevorderd)
- WebStorm (gevorderd)
- https://CodePen.io (basic, online)
- Iedere andere tekst editor zal (in principe) voldoen

Introductie HTML

- HTML staat voor Hyper Text Markup Language
- Taal waarin webpagina's worden geschreven
- Bestaat uit codes die we aan tekst kunnen toevoegen (markup betekent verrijken). Dit worden dan voornamelijk 'tags' **(DEMO).**
- Een (elke!) browser zorgt ervoor dat HTML-tekst wordt geïnterpreteerd naar een webpagina
- Huidige versie is 5.0. Met focus op duidelijk onderscheid tussen structuur en opmaak

Introductie 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. https://www.w3.org/

Een praktische website voor dagelijks gebruik is https://developer.mozilla.org/nl/ bekijk deze als je iets wilt uitzoeken m.b.t. een front-end web taal.

Structuur

- <!DOCTYPE html> versus ‘vroeger’ en gebruik van DTD Transitional (loose/strict)

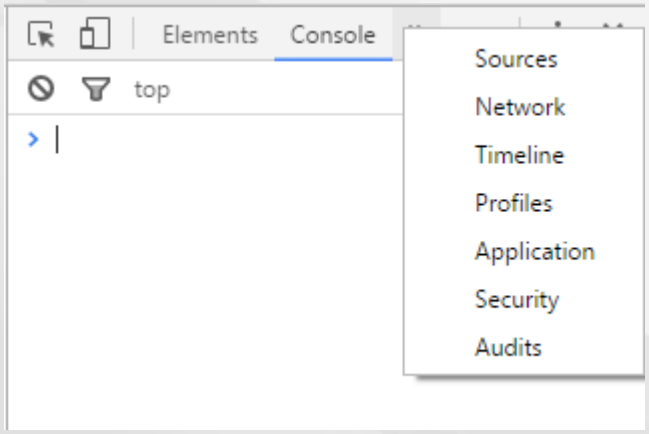
Opdracht:

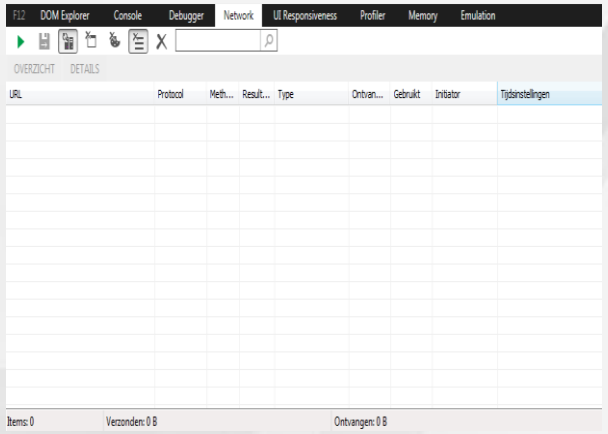
Open de Google Chrome browser (het voorbeeld werkt in een willekeurige browser) en navigeer naar een website naar keuze. Klik vervolgens met de rechter muisknop op: 'paginabron weergeven' of <ctrl><u> in Google Chrome.

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 dat er meer talen dan enkel HTML gebruikt worden bij de bouw van een webpagina.

Debugging

- Foutopsporing voor met name CSS en JavaScript code
- 'Hulpprogramma's voor ontwikkelaars‘:
  - Google Chrome: <ctrl><shift><i>, óf <f12>
  - Internet Explorwer: <f12>
  - Opera: <ctrl><shift><i>
  - FireFox: gebruik bijvoorbeeld de extensie ‘FireBug’





Vragen?

H2 – Elementen en attributen

Leerdoelen

- 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

H2 – Elementen en attributen

1 Opbouw van een HTML document

2 Blok- en inline elementen

3 Nesten van elementen

4 HTML 5 semantische structuur elementen

5 Attributen

Opbouw van een HTML document

Opbouw basale webpagina volgens W3C HTML 5 standaarden:

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

Opbouw van een HTML document

- Elementen zijn in HTML beschreven met behulp van tags.
- Een tag wordt geopend met een < en afgesloten met een >. Veel elementen kunnen ook inhoud bevatten.
- Voorbeelden van tags (en daarmee elementen) zijn:

```
<body>
<div>
<p>
```

- Iedere tag dient ook afgesloten te worden. Dit is bij de meeste tags middels een gelijknamige tag, met een backslash erin. Respectievelijk: `</body>`, `</p>` en `</div>`.
- Elementen zonder inhoud worden in de openingstag ook afgesloten. Zoals: `<input />` en `<br/>`.

Opbouw van een HTML document

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.

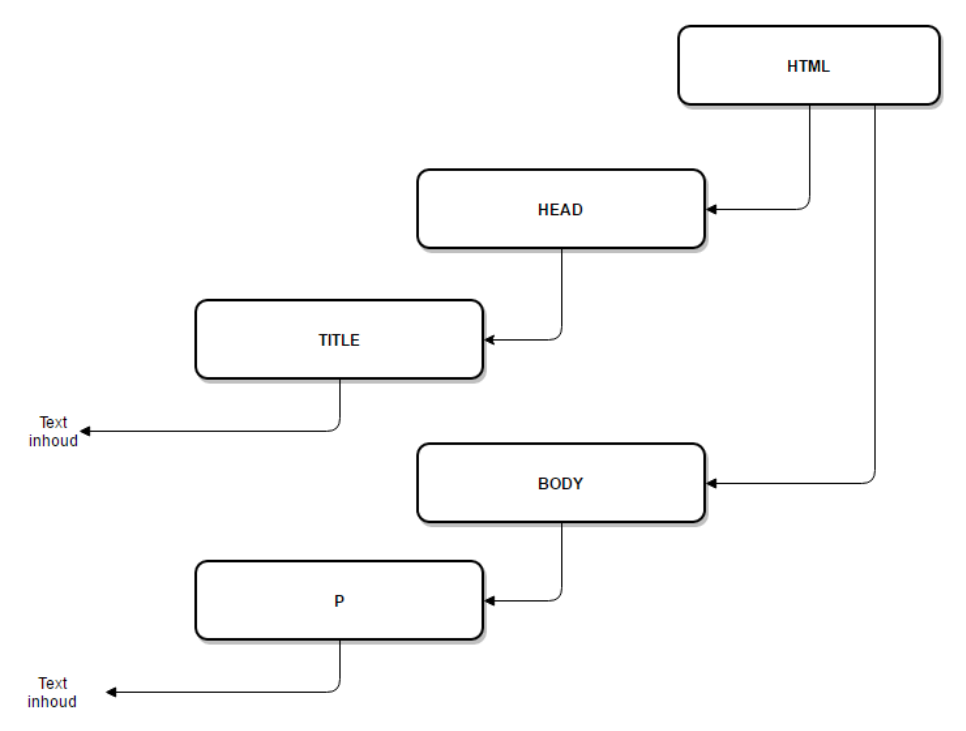
- **<head>** Hierin staan de titel van de pagina, informatie over de pagina, eventueel stijlen en eventueel JavaScript code (hierover later meer). De header wordt opgenomen tussen de tags <head> en </head>.

- **<title>** De titel van de pagina wordt opgenomen tussen de tags <title> en </title> en komt in de titelbalk van het huidige tabblad 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>.

Opbouw van een HTML document

Een webpagina is schematisch goed weer te geven omdat elementen binnen andere elementen geplaatst mogen worden:



Opbouw van een HTML document

Opdracht:

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.

Bekijk nu of u al enkele elementen van de voorgaande sheets herkent. Ziet u ook al een hiërarchie?

Blok- en inline elementen

Elementen die we in de body van de pagina gebruiken, kunnen grofweg in twee structurele groepen worden verdeeld:

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:
`<div><p><form><table><ol><ul><h1>` (t/m) `<h6>`

(DEMO + uitleg elementen)

Blok- en inline elementen

Elementen die we in de body van de pagina gebruiken, kunnen grofweg in twee structurele groepen worden verdeeld:

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: `<span><a><img />`

(DEMO + uitleg elementen)

Blok- en inline elementen

Elementen in het algemeen kunnen ook ingedeeld worden naar wat de inhoud ervan mag zijn:

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.

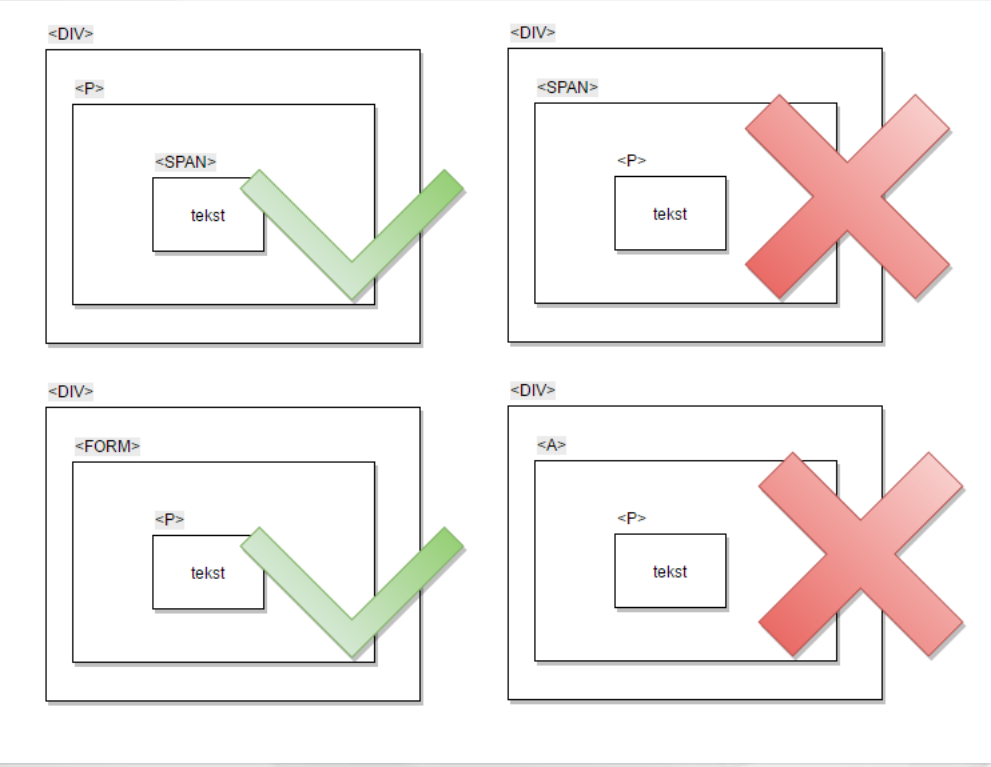
Denk aan: `<img />` en `<br />`

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: `<div>, <p>` en `<span>`

Nesten van elementen

Een schematisch voorbeeld van welke elementen wél, en welke elementen niet binnen elkaar gebruikt mogen worden:



Nesten van elementen

Tenslotte nog enkele punten met betrekking tot de lay-out:

- Lege regels in de HTML-code hebben geen resultaat en worden niet verwerkt in de browser. Ze kunnen enkel de structuur in het document verduidelijken.
- Het afbreken van de regel vindt daarom plaats met behulp van element
 en niet door in het document een return (enter) in te voeren op de gewenste plaats in de tekst.
- Meerdere opeenvolgende spaties worden in de browser ook gereduceerd tot één spatie.
- Tekst gebruikt standaard de volle breedte van de pagina.

HTML 5 semantische structuur elementen

Deze zeven nieuwe elementen vallen niet onder een categorie block of inline, omdat ze visueel bijna geen effect hebben op de pagina. Ze dienen puur een semantisch doel:

- <main> Ten behoeve van vervanging veel gebruikte die als 'hoofd' element werden gebruikt.

Let op:

Er mag maar één main element op een pagina zijn. Tevens mag deze géén onderdeel zijn van één van de hierna volgende zes elementen.

HTML 5 semantische structuur 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.

- <section> Deze tag geeft aan dat elementen die hierin gezet worden bij elkaar horen of minstens gerelateerd zijn aan elkaar. Bijvoorbeeld kopjes, paragrafen en een afbeelding over het thema treinen. Zie het als een div, maar dan met deze beschreven betekenis.

HTML 5 semantische structuur elementen

- <header> Dient niet verward dient te worden met het <head> element. <header> kan meermaals binnen een webpagina worden gebruikt om inleidende informatie te geven over bijvoorbeeld een hele pagina, een sectie of een artikel.

- <footer> 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.

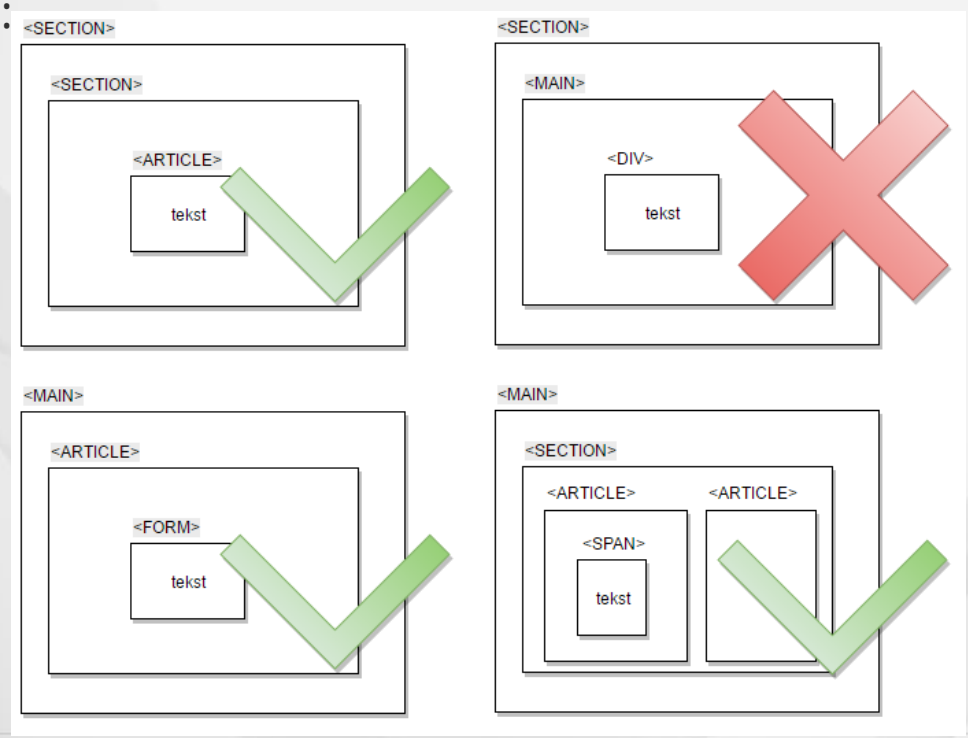
HTML 5 semantische structuur elementen

- <aside> Bedoeld om een balk aan de zijkant van de webpagina mee te specificeren (contactinformatie, wordcloud, etc).

- <nav> Hierin kunnen wij 'blokken' met navigatie links kwijt. Bedoeld om navigatie-delen van een webpagina te beschrijven. Dus niet iedere link op een pagina hoort in een <nav> tag thuis.

HTML 5 semantische structuur elementen

Een schematisch voorbeeld van welke elementen wél, en welke elementen niet binnen elkaar gebruikt mogen worden:



HTML 5 semantische structuur elementen

Opdracht:

Schrijf een eenvoudige HTML webpagina over een willekeurig thema dat u aanspreekt. Gebruik daarin minimaal: een article, een header, een footer en een nav element. Geef de pagina ook een toepasselijke *title*.

Test ook of het werkt door je webpagina op te slaan als .html document en te openen, of maak hem op 'https://CodePen.IO'.

Attributen

- Attributen zijn eigenschappen met eventueel een waarde, die onderdeel zijn van een element.
- De karakteristieken van een element kunnen hiermee nog nauwkeuriger worden bepaald.
- Attributen worden alleen in de begintag van een element opgenomen.
- Bij meerdere attributen per element is de volgorde niet van belang.
- Sommige elementen hebben helemaal geen attributen (zoals `<b>`, `<i>` en `<sub>`).

Voorbeeld attribuut: `<input type="text" />`

Attributen

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, niet verplicht) tussen dubbele quotes opgenomen.
- Attribuut en attribuutwaarde worden gescheiden door een is-gelijk-teken (=).

Attributen

We kennen in HTML 5 ook een aantal generieke attributen, waarvan we er enkelen hier zullen toelichten:

- **Accesskey**: toetsencombinatie naar het element
- **Class**: om elementen te categoriseren voor CSS en JavaScript
- **Contenteditable**: mag de inhoud gewijzigd worden?
- **Contextmenu**: kunstmatig rechtermuis menu op basis van <menu> en <menuitems> elementen (FireFox)
- **Data-***: content doorsturen ipv met 'hidden fields'. Veel gebruikt door JavaScript en server talen.

Attributen

- **Draggable**: is een element versleepbaar?
- **Hidden**: is het element (op dit moment) zichtbaar?
- **Id**: variant op het veel gebruikte ‘class’ element. Zeer toepasbaar indien we een *unieke* naam of referentie naar een element willen maken.
- **Style**: hiermee voegen we CSS stijlen toe aan het element.
- **Tabindex**: een numerieke waarde waarmee wij de zogeheten ‘tabvolgorde’ kunnen regelen.

(DEMO)

Attributen

- **Title**: extra informatie over een element voor zoekmachines
- **Translate**: willen wij de inhoud van dit element kunnen (laten) vertalen?
- **Dir**: met de waarde ‘rtl’ geven we ‘right-to-left’ tekst richting aan.
- **Lang**: geeft de taal van het element aan (fr, nl, de, etc)
- **Spellcheck**: mag dit element mee in spellingscontrole?
- **Dropzone:** (nog geen ondersteuning voor)

Vragen?

Opdracht

- Doorloop de opdrachten op mijn.vijfhart.nl

H3 - Opmaak

Leerdoelen

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

H3 - Opmaak

1 Fysieke opmaak elementen

2 Semantische opmaak elementen

3 Webpagina’s opmaken met Cascading Style Sheets (CSS)

Fysieke opmaak elementen

Fysieke opmaak elementen zeggen direct iets over de tekst die binnen de bijbehorende tags staat. Opmaak wordt met de komst van HTML 5 aangeraden door CSS te laten doen.

- Het wordt afgeraden fysieke opmaak elementen te gebruiken.
- Voorbeelden zijn: `<b><i><u>` en `<br/>`.
- Het regelovergang element is als enige (nog?) niet noodzakelijk vervangen door opmaak van CSS.

Semantische opmaak elementen

In tegenstelling tot de fysieke opmaak elementen, wordt met HTML 5 sterk aanbevolen waar nodig semantische opmaak elementen te gebruiken:

- 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.

Semantische opmaak elementen

- <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.

```
<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>
```

Semantische opmaak elementen

- <h1> t/m <h6> kopjes. Als onderscheid tussen titels.
- <code> Wordt veel op fora en websites zoals *Stackoverflow* en *Experts-exchange* gebruikt, om te tonen dat iets om een code voorbeeld gaat (zorgt voor overzichtelijkheid).
- <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 na klik voor een uitklapbaar stukje tekst
- <time> In een time element kan een tijdstip worden geplaatst. Deze tijd dient in het bijbehorende optionele attribuut 'datetime' geplaatst te worden, volgens een vaste opmaak.

Commentaar in HTML

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>
```

Webpagina's opmaken met Cascading Style Sheets (CSS)

- Webpagina's opmaken doen we middels CSS
- Met CSS 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.

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

Webpagina's opmaken met Cascading Style Sheets (CSS)

- Gangbaar is dat CSS code in een aparte .css file wordt opgeslagen.
- Vanuit de HTML pagina wordt verwezen naar een .css document.
- Het mogelijk meerdere .css files aan een document te koppelen (twee voor weergave op PC, één voor tablet weergave en één voor print opmaak).

Webpagina's opmaken met Cascading Style Sheets (CSS)

Manieren om CSS code op je HTML document toe te passen:

- **Embedded**: 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.

  Voorbeeld:

```
<style type="text/css">
   p { color: green; }
</style>
```

Webpagina's opmaken met Cascading Style Sheets (CSS)

Manieren om CSS code op je HTML document toe te passen:

- **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.

  Voorbeeld:

```
<p style="color: green">
```

Webpagina's opmaken met Cascading Style Sheets (CSS)

Manieren om CSS code op je HTML document toe te passen:

- **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"/>
```

Webpagina's opmaken met Cascading Style Sheets (CSS)

Manieren om CSS code op je HTML document toe te passen:

- **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>
```

Webpagina's opmaken met Cascading Style Sheets (CSS)

Tot slot:

- CSS is een aparte programmeertaal, maar is erg handig om als HTML programmeur te kennen.
- Wij bieden hier uiteraard cursussen voor aan, maar bekijk (bijvoorbeeld) eerst eens:

  https://www.w3schools.com/css/

Vragen?

Opdracht

- Doorloop de opdrachten op mijn.vijfhart.nl

H4 – Images, hyperlinks en embedding

Leerdoelen

- 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

H4 – Images, hyperlinks en embedding

1 Afbeeldingen met het img element

2 Gebruik van het figure element

3 Opkomst van het picture element

4 Imagemaps

5 Hyperlinks

6 Hyperlinks naar ankers

7 Embedding en iFrames

Afbeeldingen met het img element

Het opnemen van plaatjes (m.b.v. de <img> tag) kan een webpagina aantrekkelijk maken.

- Is een inline element, dus er kan tekst naast getypt worden.
- Mag in tegenstelling tot andere inline elementen zelf een breedte en hoogte hebben (width en height).
- Heeft veel attributen, waarvan de belangrijksten zijn:

  - **src:** de bron van de afbeelding; een URL of pad.
  - **alt:** tekstweergave indien afbeelding niet toonbaar
  - **title:** tekst bij 'mouse over' en t.b.v. zoekmachines
  - **width** en **height**

Afbeeldingen met het img element

Voorbeeld van een <img> element:

```
<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 en <img /> tag achter de schermen letterlijk vervangen door de *bit representatie* van een fysieke afbeelding.

Afbeeldingen met het img element

Voor de bron (het src attribuut) kunnen wij kiezen voor het gebruik van een absoluut pad of relatief pad:

- **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 wilt dat zeggen: het gehele pad, gezien vanaf de root.

  Bijv:
  `https://upload.wikimedia.org/wikipedia/en/d/d7/Tingle.png`
  Bijv: `c:\games\zelda\tingle.jpg` (deze is onveilig, is niet te gebruiken)

Afbeeldingen met het img element

- **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.

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

Afbeeldingen met het img element

Bij zware en veel bezochte websites kan het interessant zijn een afbeelding ‘inline’ in het document op te slaan middels een base64 encoding.

Dit valt echter buiten de scope van deze cursus. Wil je hier toch mee oefenen? Bekijk dan:

http://base64online.org/

Gebruik van het figure element

- Met <figure> geven wij semantisch aan dat er een afbeelding (foto, tekening, enzovoorts) in dit element zit.

- Binnen het <figure> element hoort nog steeds een ‘normale’ <img /> tag met het src attribuut.

- Aangeraden wordt om een <figcaption> element binnen het <figure> te gebruiken om een tekst aan te geven die bij de afbeelding verschijnt.

Opkomst van het picture element

- Biedt de mogelijkheid apparaat en resolutie afhankelijke afbeeldingen te laden
- Komt ten goede van performance (download enkel het de juiste afbeelding, down- of upscaling niet nodig
- Bevat één of meer <source> elementen met een ‘srcset’ en bij voorkeur ‘media’ attribuut
- Bevat alsnog een verplicht <img> element

Demo: https://codepen.io/5hart/pen/qoRMQE

Imagemaps

Staat voor: het opdelen in sectoren en de browser anders laten reageren op het klikken op verschillende delen van de afbeelding.

Nodig om imagemap te maken:

- 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 vormen kunnen zijn een: rect, circle of poly.

Imagemaps

Voorbeeld:

```
<map name="sectoren">
    <area shape=rect coords="0,0 144,73"
    HREF="alert('blauw');" alt="blauw" />
    <area shape=rect coords="145,74 306,142"
    HREF="alert('rood');" alt="rood" />
    <area shape=rect coords="145,0 306,73"
    HREF="alert('grijs');" alt="grijs" />
</map>
<img src="map_plaatje.jpg" ismap alt="image map
voorbeeld" usemap="#sectoren"/>
```

Hyperlinks

Een (hyper)link is klikbare content waar een actie volgt op het klikken.

- Het element voor een link is <a>
- Het attribuut ‘href’ dient aanwezig te zijn en dient als waarde het pad (absoluut of relatief) te krijgen naar de doelpagina.
- De tekst of elementen tussen de <a> en tags is wat de gebruiker ziet en waar hij of zij op kan klikken.

Hyperlinks

Voorbeelden:

Ga naar Google

Klik hier om verder te gaan...

Deze link doet niets

Hyperlinks

In een hyperlink href attribuut, kan ook een e-mail adres staan met ‘mailto:’ ervoor. Dit wordt sterk afgeraden omdat er hierdoor een e-mail adres in je HTML code benoemd wordt.

Wat zou het gevaar hiervan kunnen zijn?

Hyperlinks naar ankers

Is een verwijzing naar een ander deel van dezelfde pagina als waar de link staat.

Demo: https://codepen.io/5hart/pen/wmgYBm

Embedding

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

- Kan meer dan alleen audio of video embedden
- Src attribuut is noodzakelijk
- Type attribuut wordt door W3C aangeraden (op basis van bekende media types

Embedding middels <audio> en <video> komt in hoofdstuk 7 aan bod.

iFrames

Frames kunnen worden gebruikt om een andere webpagina in te laden binnen de huidige pagina.

- Wordt niet aangeraden wegens:
  - Performance
  - Looks
  - Loss of control

Alternatief: iFrame:

```
<iframe frameborder="0" scrolling="no"
style="border: 0px;" src="url_naar_het_boek"
width="500" height="500"></iframe>
```

Vragen?

Opdracht

- Doorloop de opdrachten op mijn.vijfhart.nl

H5 – Tabellen, lijsten en speciale karakters

Leerdoelen

- 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

H5 – Tabellen, lijsten en speciale karakters

1 Tabel onderdelen

2 Tekst over kolommen of rijen verspreiden

3 Eigenschappen groeperen

4 Lijsten

5 Definitielijsten

6 Speciale tekens

Tabel onderdelen

Bekijk de onderstaande video over de opbouw en het maken van tabellen:

https://youtu.be/WACK1Alxt3M

De opbouw van een tabel in een voorbeeld uitgewerkt:

https://codepen.io/5hart/pen/BrpqwO

Tekst over kolommen of rijen verspreiden

- Over tekst over kolommen te verspreiden kunnen we aan de betreffende kolom (td) het attribuut ‘colspan’ meegeven en die op de nodige waarde instellen.

- Om tekst over meerdere rijen te verspreiden kunnen we aan de betreffende rij (tr) het attribuut ‘rowspan’ meegeven en die op de nodige waarde instellen.

Tekst over kolommen of rijen verspreiden

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

Demo: https://codepen.io/5hart/pen/qoWzGv

Toelichting op bovenstaande:

- 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").

Eigenschappen groeperen

- Kolommen groeperen. Kenmerken gelden voor alle kolommen in deze groep.
- Binnen een <colgroup> element kunnen <col> elementen staan, met ieder een eigen opmaak
- Het enig in HTML 5 bruikbare attribuut is: *span*. Deze geeft aan hoeveel naast elkaar gelegen kolommen tot de kolomgroep behoren (default waarde is 1).
- De opmaak wordt vrijwel altijd gedaan middels CSS.

Demo: https://codepen.io/5hart/pen/Brpqrq

Lijsten

- Gegevens kunnen we in opsommingen weergeven met behulp van de elementen <ol> (geordend) en <ul> (ongeordend).
- Beide soorten lijsten bevatten <li> element die de daadwerkelijke content van de lijsten voorstellen.
- Middels het optionele *type* attribuut kunnen wij aangeven welk symbool er voor de opsomming gebruikt wordt. Bijv: d*isc* (dicht rondje) en *circle* (open rondje) voor een <ul> en 1 (1,2,3), a (a,b,c), A (A,B,C), i (i,ii,iii), I (I,II,III) voor een <ol>.

Lijsten

- Elk ul en ol element hoort één of meer li elementen te bevatten.
- 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.

Demo: https://codepen.io/5hart/pen/wmgYER

Definitielijsten

- Een definitielijst is een lijst van termen
- Vervolgens wordt deze betreffende term uitgelegd
- Wordt aangemaakt met het <dl> element
- Binnen een <dl> maken we gebruik van <dt> en <dd> elementen
- Een <dt> element benoemt een term in een lijst
- Een <dd> element beschrijft een term in een lijst

Demo: https://codepen.io/5hart/pen/bvgmOZ

Speciale tekens

In moderne browsers gaat het weergeven van tekens vaak wel goed, maar in oudere browsers is het gebruik van character entities een must.

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

Demo: https://codepen.io/5hart/pen/gegQpq

Vragen?

Opdracht

- Doorloop de opdrachten op mijn.vijfhart.nl

H6 - Formulieren

Leerdoelen

- 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

H6 - Formulieren

1 Het form element

2 Tekstvelden en keuzelijsten

3 Input element

4 Gegevens groeperen

5 HTML 5 elementen

6 HTML 5 element attributen

Introductie tot formulieren

Ga naar:

https://www.vijfhart.nl/opleidingen/ en blader naar de categorie ‘software development’.

Bekijk het formulier, óók door de bron van de pagina door te nemen.

Het form element

- Definieert het begin en het einde van een formulier
- Het element kan vier attributen bevatten: *method*, *action*, *autocomplete* en *novalidate*.

De attributen '*method*' en '*action*' zijn veruit de twee belangrijkste en zullen we nu bespreken.

Het form element – method attr.

- 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.
- De input is eenvoudig af te handelen omdat alle invoervelden vermeld zullen staan in de adresbalk. 'Post' is in veel gevallen te aan te raden verzend methode.

Het form element – action attr.

- Geeft aan waar (welk bestand) de informatie uit het formulier heen gestuurd moet worden.
- Bestemming is meestal een script of programma op de server dat de informatie uit het formulier moet verwerken. Denk aan een file ‘verwerk.php’
- De informatie van het formulier kan ook zonder tussenkomst van een script of programma direct naar een opgegeven e-mailadres worden gestuurd.

Tekstvelden

Een tekstveld maken we middels het <textarea> element. Dit heeft een ‘rows’ en een ‘cols’ attribuut:

- Rows bepaalt hoe hoog het tekstvak is, uitgedrukt in het aantal rijen (regels) tekst (los v/d input v/d user).
- Cols bepaalt hoe breed het tekstvak is, uitgedrukt in het aantal kolommen (karakters) tekst.

DEMO: https://codepen.io/5hart/pen/YaNRGO

Keuzelijsten

Een keuzelijst maken we middels het <select> element.

- Verschillende opties maken we middels <option> elementen (binnen het <select> element
- Er kan een default geselecteerde waarde ingesteld worden middels het *selected* attribuut van een <option> element

DEMO: https://codepen.io/5hart/pen/rdjQLy

Input element

Met dit element wordt een veld gedefinieerd, waarin de gebruiker gegevens kan invoeren. Soort is afhankelijk v/d '*type*' attribuut waarde:

- "text": tekstgegevens (al of niet verborgen)
- "checkbox": een selectie m.b.v. aankruisvakjes
- "radio": een selectie m.b.v. keuzerondjes
- "submit": opdracht tot het verzenden van ingevoerde informatie
- "reset": het herstellen van de veldwaarden naar de defaults
- "color": kleuren kiezer
- "range": range kiezer
- "date": datum kiezer
- "file": bestand kiezer

DEMO's: https://codepen.io/5hart/pen/YaNRGO

https://codepen.io/5hart/pen/Koaraz

Gegevens groeperen

- 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.

- Het <legend> element zorgt ervoor dat er een bijschrift bij de door fieldset gegroepeerde controls wordt weergegeven.

- Binnen het <fieldset> element moet als eerste het legend element worden opgenomen. Daarna volgen de andere elementen.

Er is ook een <label> element met for="" attribuut 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

HTML 5 elementen - getallen

Om een getal op te kunnen geven middels een formulier kan o.a. gebruik worden gemaakt van de volgende input typen:

- type = “text”
- type = “number”
- type = “range”

DEMO: https://codepen.io/5hart/pen/MVJZaQ

HTML 5 elementen - data

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

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

Opdracht: Maak zelf een webpagina met daarop minimaal twéé van de bovenstaande input typen.

HTML 5 elementen - overig

- Datalist (met daarbinnen <option> elementen)
- color (input)
- email (input)
- search (input) -- enkel semantisch!
- url (input)

DEMO: https://codepen.io/5hart/pen/wmgRog

HTML 5 elementen - progress

- De <progress> tag doet dienst als progressiemeter.
- 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.

DEMO: https://codepen.io/5hart/pen/zWNywy

HTML 5 elementen - meter

- Vaak wordt het <meter> element gebruikt als alternatief voor een progress element. Daar is hij semantisch niet voor bedoeld.

- Is bedoeld 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).

- Enkel het value attribuut is verplicht. De meter moet een waarde hebben.

Voor meer informatie over de attributen: MDN website.

HTML 5 element attributen

Onderstaand volgt een overzicht met alle overige HTML 5 attributen die op form elementen van toepassing zijn, In willekeurige volgorde:

Disabled, maxlength, readonly

Size, value, autocomplete

Autofocus, form, Formaction

Height, width, list

Min, max, multiple

pattern (regexp), placeholder, required, step

DEMO: https://codepen.io/5hart/pen/PRWXJb

HTML 5 element attributen

Attributen die alleen 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. te worden.

- Novalidate

De aanwezigheid van dit attribuut zonder waarde is al voldoende om hem aan te zetten. 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, 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).

Vragen?

Opdracht

- Doorloop de opdrachten op mijn.vijfhart.nl

H7 – Audio en video

Leerdoelen

- 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

H7 – Audio en video

1 Het video element

2 Het audio element

3 Het track element

Inspiratie sessie

Bekijk deze leuke webpagina die gebruikt maakt van de <video> en <audio> elementen:

Art Copy Code

Het video element

Met behulp van de <video> tag kunnen wij een video zoals een opname of animatie tonen aan de webpagina bezoeker.

Mogelijke instellingen te zetten voor:

- 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

Om een video af te laten spelen dient:

- autoplay aan te staan, óf
- knoppenbalk aanwezig te zijn, óf
- JavaScript code aanwezig te zijn om de video te kunnen starten

Het video element

Met behulp van de <video> tag kunnen wij een video zoals een opname of animatie tonen aan de webpagina bezoeker:

DEMO

Mogelijke instellingen te zetten voor:

- 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

Om een video af te laten spelen dient:

- autoplay aan te staan, óf
- knoppenbalk aanwezig te zijn, óf
- JavaScript code aanwezig te zijn om de video te kunnen starten

Het video element

Aanschouw de youtube pagina van Izzy Swan, een meubelmaker. Hij maakt video opnames van zijn projecten en geeft online les in het maken van interieurstukken.

Omdat Izzy zijn video aantallen snel toenemen, heeft ook hij de behoefte aan meer overzicht op zijn pagina, zodat de bezoekers sneller de gewenste video's kunnen bekijken.

Wil je deze zogeheten ‘frontjes’ zelf realiseren buiten Youtube, maar wel met HTML 5? Gebruik dan het *poster* attribuut van de <video> tag.

Bekijk hier onder het kopje 'poster', de uitleg van het poster attribuut op de MDN website.

Het audio element

Met behulp van de <audio> tag kunnen wij een audiospeler genereren, zoals <video> dit doet voor het afspelen van videofragmenten.

DEMO: https://codepen.io/5hart/pen/OzjYjx

Bekijk de MDN website voor informatie over file typen.

De belangijkste attributen:

- Controls: wel of geen besturingselementen zichtbaar?
- Autoplay: moet de audiofile direct starten zodra geladen?
- Loop: Moet de opname herhaaldelijk afgespeeld worden?
- Src: wat is de filename en e.v.t. het pad van de audiofile?
- Preload: dient het audio bestand alvast ingeladen te worden voordat het afgespeeld kan worden?

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, screenreader tekst, hoofdstuk aanduider 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). De belangrijkste noemenswaardige attributen hiervan zijn:

- kind (meestal ‘subtitle’)
- label
- src
- srclang

Het track element

Een WebVTT bestand bevat cue's met inhoud waar het 'src' attribuut van een <track> element naar verwijst. Voorbeeld van een WebVTT bestand:

```
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 <strong>fireworks</strong>. The lights on the party tree.
```

Het track element

Ieder blok met tijdsspanne en content is een ‘cue’. Een cue bevat zoals in het vorige voorbeeld vaak een tekst, maar mag bijvoorbeeld ook HTML en zelfs JSON code bevatten. Dat is een JavaScript object notatie zoals:

```
{
    “title”: “The Matrix”,
    “short_description”: “Taking the red pil”
}
```

Vragen?

Opdracht

- Maak (naar keuze) een audio of video element met een via google gevonden audio of video bestand
- Kijk of je er ook de bijbehorende ‘controls’ bij kunt tonen

- Wil je hem wel of niet automatisch laten starten?

H8 – Canvas en SVG

Leerdoelen

- Een canvas kunnen maken en daar een figuur op kunnen plaatsen met HTML
- Weten welke mogelijkheden er zijn om een SVG afbeelding op een webpagina te plaatsen en minimaal één van deze mogelijkheden kunnen toepassen

H8 – Canvas en SVG

1 Het canvas element

2 SVG afbeeldingen

Het canvas element - Inspiratie

Mooi voorbeeld van het gebruik van het HTML canvas:

EffectGames.com

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.

Voor heel complexe tekeningen, is natuurlijk een gezonde hoeveelheid tekenvaardigheid nodig.

Het canvas element

Onderstaand voorbeeld beschrijft de verschillende mogelijke vormen, tekst en afbeeldingen die wij kunnen verwerken op een pagina.

Er is bijna alleen maar JavaScript code nodig om te kunnen werken met een canvas. Wil je dit element echt kunnen doorgronden, is het raadzaam een basiscursus JavaScript te volgen.

DEMO: https://codepen.io/5hart/pen/GyvbNG

SVG afbeeldingen

Wat is SVG:

- Een bestandsformaat waarmee afbeeldingen kunnen worden weergegeven
- Opgebouwd uit een aantal coördinaten, kleuren en vullingen
- Vastgelegd zijn met behulp van XML tags

Voordelen:

- 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

Nadelen:

- Door complexiteit lastig bij ingewikkelde afbeeldingen

SVG afbeeldingen

Voorbeelden van SVG output:

Onderstaand volgt een voorbeeld met toevoegen van een SVG bestand aan je webpagina:

DEMO

SVG afbeeldingen

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

DEMO

Vragen?

Opdracht

- Probeer aan de hand van ons tweede SVG voorbeeld, de grafiek naar eigen wens wat aan te passen.

- Wijzig bijvoorbeeld de getallen en teksten

- Bonus: probeer de grafiek te ‘kantelen’; maak de grafiek lijnen verticaal i.p.v. horizontaal.

H9 – HTML 5 API’s

Leerdoelen

- 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 verbeteren door langlopende processen te laten behandelen door web workers.

H9 – HTML 5 API's

1 Inleiding

2 Offline applicaties

3 Usability API's

4 Mobiele device API's

5 WebSockets

6 Web workers

Inleiding

Wat zijn API's?

- Application Programming Interface's.
- Stukken programmatuur die gemaakt zijn om functionaliteit toe te voegen aan een systeem.

Algemeen:

- Binnen HTML 5 continu in ontwikkeling
- Veelal JavaScript kennis vereist

Offline applicaties

Om te voorzien in de groeiende behoefte aan grotere en veelzijdigere online applicaties, zijn er met HTML 5 een aantal erg praktische features bij gekomen ten behoeve van client-side opslag (en performance verbetering):

- application cache
- localStorage
- web SQL & indexed database

We gaan deze drie technieken nu bespreken

Offline applicaties – application cache

Wat is caching van bestanden?

- Opslaan van bestanden lokaal bij de eindgebruiker
- Bestand hoeft later niet meer gedownload te worden
- Komt vaak zeer ten goede van de performance

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

Offline applicaties

Waar staan de files (of zelf niet leesbare containers!) opgeslagen?

Hier

Welke files worden gecached? Dat staat in de manifest file:

- Welke files gecached dienen te worden (*cache* categorie)
- Welke files in geval van verlies van de connectie (*fallback* categorie)
- welke files er in ieder geval *niet* via de cache mogen lopen indien er verbinding is met een website (*network* categorie)

DEMO (voorbeeld content)

Offline applicaties – local storage

Met local storage, bedoelen wij:

- lokaal opslaan van gegevens voor later gebruik.
- Veiliger dan cookies
- Meer ruimte dan cookies
- Opslag middels key-value paren

Denk bijvoorbeeld aan:

- BSN nummer - Naam
- Barcode - Titel
- Voornaam - Geboortedatum
- Kantoornummer - Kantoornaam

Offline applicaties – local storage

Twee implementaties van local storage:

sessionStorage, dat is lokale opslag, die verloren gaat zodra de eindgebruiker de browser afsluit.

Voordeel: gevoele informatie blijft niet altijd behouden op de computer

Implementatie: middels window.sessionStorage JavaScript object.

Offline applicaties – local storage

Twee implementaties van local storage:

localStorage, dat is eveneens lokale opslag, die behouden blijft wanneer de browser wordt afgesloten en/of heropend.

Voordeel: winkelwagen, gebruikersprofielen, etc blijft beschikbaar, zelfs bij na (bijv.) 2 weken wederom bezoeken van de pagina.

Implementatie: middels window.localStorage JavaScript object.

Offline applicaties – local storage

Bekijk hier een voorbeeld met sessionStorage en localStorage:

https://codepen.io/5hart/pen/zpdgEj

https://codepen.io/5hart/pen/yKgGxx

Usability API's

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 door de WebDB specifieke SQL taal.

IndexedDB slaat gegevens op als objecten, maakt gebruik van NoSQL database; vaak sneller voor veel werken met JavaScript objecten. Voor meer info: IndexedDB.

* Bekijken inhoud kan ook onder de developer tools

Usability API's

We gaan nu kijken naar twee API's die de gebruiksvriendelijkheid van een webpagina kunnen verbeteren, namelijk met de:

- Page visibility API
- Fullscreen API

Usability API's – Page visibility API

Dit gaat om de mogelijkheid tot het afvangen van het moment dat een gebruiker naar een andere pagina wilt navigeren.

Dit doen we middels een JavaScript event genaamd:

- webkitvisiblilitychange (in Chrome)
- msvisibilitychange (in IE)
- visibilitychange (in Opera en FireFox).

Toepassing voorbeeld:

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").

DEMO (bekijk de document.title)

Usability API's – 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).

- Hiervoor gebruiken wij de requestFullscreen() JavaScript functie
- Er wordt dan aan de gebruiker gevraagd of hij / zij wilt overschakelen naar een volledig-scherm weergave
- Programmeertechnisch afsluiten ervan kan ook door de exitFullscreen() functie

Usability API's – 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).

- Hiervoor gebruiken wij de requestFullscreen() JavaScript functie
- Er wordt dan aan de gebruiker gevraagd of hij / zij wilt overschakelen naar een volledig-scherm weergave
- Programmeertechnisch afsluiten ervan kan ook door de exitFullscreen() functie

DEMO

Mobiele device API's - Geolocation

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)

I.v.m. veel JavaScript code, zullen we dit verder niet behandelen in deze cursus. Voor meer informatie hierover: MDN.

Mobiele device API's – File API

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 verificatie bij een BitCoin broker
- 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'. 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.

DEMO

Mobiele device API's – JS objecten

Binnen een browser zijn allerlei objecten gecreëerd 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 JavaScript kennis aan te bevelen.

Mobiele device API's – Vibrate API

De uitwerking van deze API kan ervoor zorgen dat het (mobiele) device waarmee de eindgebruiker de webpagina benadert, begint te trillen.

- Navigator object heeft vibrate() functie gekregen. Daaraan kunnen wij optioneel meegeven hoe lang het device dient te trillen in milliseconden (1000 milliseconden = 1 seconde).
- Een API die vooral binnen gaming zijn toepassing vindt

DEMO (mobiel!)

Mobiele device API's – getUserMedia API

Om gebruik te kunnen maken van media apparaten op het device van de eindgebruiker, kunnen wij de getUserMedia API gebruiken.

- Vraagt de gebruiker of wij gebruik mogen maken van zijn of haar camera en/of microfoon.
- Mogelijkheid tot 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).

Mobiele device API's – Batterylevel 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.

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.

Om WebSockets 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.

WebSockets

Code om een WebSocket aan te maken:

//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');
```

WebSockets

Code om een WebSocket aan te maken:

//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’);
```

Dit volgende voorbeeld gebruikt een bestaande 'oefen' websocket server: jsFiddle Marko Draisma.

Web workers

Web workers zijn werktaken die wij kunnen starten, en parallel kunnen laten lopen aan de rest van de webpagina.

Bijvoorbeeld:

- 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

Web workers

Voor dit voorbeeld 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:

DEMO: https://codepen.io/5hart/pen/gegqYx

(dit voorbeeld werkt het best op een webserver)

Vragen?

Einde