Aantekeningen bij The Complete Web Developer Course

**Deel 1 HTML**

**Les 5 – Your control panel**

Ik moet index.html in Safari expliciet opgeven om de gewijzigde webpagina te laten zien, anders krijg ik de default te zien.

**Les 6 – Uploading files with ftp**

FTP staat default dicht bij ecoWebhosting. Deze moet unlocked worden voor liefst niet te lang. Als je het getoonde domein niet hebt geregistreerd, dan moet je het getoonde FTP Server IP adres gebruiken.

Rob gebruikt de FireFTP plugin van FireFox, ik heb FileZilla geinstalleerd. Voordeel van FireFTP is dat je de code direct op de hosting site kunt editen, zonder deze te moeten downloaden. FileZilla kan dat niet zo te zien.

De ftp accountgegevens staan in het FTP gedeelte van het CPanel!

Als oefening even index.html uit de public dir aanpassen en vervolgens jouw site (famasoft.com) bezoeken vanaf het CPanel (rechts onder Account Info ‘visit website’ klikken).

**Les 7 – Getting help**

Het toevoegen van een link naar je webcode in geval je hulp nodig hebt. Gebruik hiervoor de gratis site jsbin.com waar je je code even naartoe kopieert en kopieer de url naar de Udemy blog van deze cursus.

**Les 8 – Hello World**

Met de gratis html editor TextWrangler een ‘helloworld.html’ maken. Ik heb eerst een folderstructuur aangemaakt in Documents: ‘Web Development’ en daaronder een folder ‘1. HTML’.

Open hiertoe een nieuw tekst document in TextWrangler. Schrijf een tekst in het window en save de file in de ‘1. HTML’ dir als helloworld.html (kleine letters bij voorkeur). Klik vanuit de finder vervolgens op de file en Safari opent een nieuw tab waarin de ingevoerde tekst te zien is.

**Les 9 – Structure of a webpage**

Om de sourcecode van een webpagina te kunnen bekijken in Safari, eerst hiervoor de Ontwikkel optie vrijgeven vanuit de Safari voorkeuren in de Geavanceerd tab. Vervolgens op de webpagina in kwestie rechtsklikken en Toon paginabron selecteren.

Rob gebruikt exapmle.com als eenvoudig voorbeeld waarin de structuur wordt uitgelegd.

Elk html document begint met een <!doctype html>, andere doctypes zijn ook beschikbaar.

Vervolgens de <html> tag: geeft aan dat de html code hier start. Dit eindigt bij de </html> closing tag.

Daarna de <head> tag, deze geeft diverse info over de pagina en links naar andere files die nodig zijn, zoals javascript. Binnen de <head> sectie:

- <title> dit is de tekst die in de tab bar staat.

- <meta> geeft info zoals lettertype, content type en viewport. Met viewport kan de breedte van de pagina ingesteld worden, bijv. gelijk aan device afmeting etc.

- <style> geeft aan hoe de elementen op het scherm weergegeven worden, bijv afgeronde hoeken, fontgrootte etc. Meer daarover in hoofdstuk 2.

Elke pagina heeft een <head> en <body> sectie. De <body> sectie geeft de inhoud van de pagina. Deze is onderverdeeld in zg. divisions <div> die ieder een stukje inhoud van de pagina beschrijven.

Binnen de <div> sectie:

- een <h1> header tag, de header tekst wordt groot en vet afgedrukt

- een <p> paragraph sectie, met daarin tekst of een hyperlink die binnen de paragraph omgeven is door een <a> en </a> tag:

<p><a href=’url’>’tekst’</a></p>,

waarbij ‘url’ de hyperlink is en ‘tekst’ de tekst is die wordt getoond en waar op geklikt kan worden.

**Les 11 – Your own example.com**

Kopieer de sourcode van example.com naar de file in TextWrangler en save deze als index.html. Deze file wordt geladen als de homepage van een website.

Het werkt het handigst als zowel TextWrangler als Safari op het scherm staan, zodat wijzigingen in index.html direct zichtbaar gemaakt kunnen worden in Safari.

Pas de <title> tekst aan en gooi de styling in de header weg. Gooi ook de content weg (<div> spul), zodat een blanco pagina over blijft.

**Les 12 – Header tags**

Voeg in de body header tags met teksten toe:

- <h1>Grootste header</h1>

- <h2>Een na grootste header</h2>

- <h3>Twee na grootste header</h3>

- <h4>Nog kleinere header</h4>

- <h5>En nog een niveau kleinere header</h5>

- <h6>En nog een niveau lager</h6>

- <h7>Test</h7>

De headers worden steeds kleiner t/m h6, alle tekst is vet. Vanaf h7 wordt normale paragraaftekst getoond (niet vet) en de header dus in feite genegeerd.

**Les 14 - Paragraphs**

Verwijder de <h7> header en voeg platte tekst toe na de laatste <h6> header. Doe hetzelfde op een nieuwe regel. Voeg na d e tekst op de tweede regel een flink aantal spaties en nieuwe tekst toe. Merk op dat html line breaks en spaties negeert: alle hier ingevoerde tekst wordt op dezelfde regel getoond met een spatie ertussen.

Om de tekst apart weer te geven moeten ze in paragraphs <p> geplaatst worden. Doe dit met de twee ingevoerde regels.:

<p>Dit is regel 1</p>

<p>Dit is regel 2</p>

Het valt op dat er een flink gat tussen de paragrafen zit. Om tekst binnen een paragraaf direct onder de eerste regel te krijgen moet een linebreak <br /> toegevoegd worden:

<p>Dit is regel 1<br />Gevolgd door een nieuwe regel binnen de paragraaf</p>

De <br /> is een zg. self closing tag. Op zich is de ‘/’ niet nodig voor ‘oude’ html. Echter voor xhtml compatibiliteit moet een self closing tag gebruikt worden.

Voeg een horizontale regel tussen beide paragrafen toe:

<hr />

**Les 16 – Formatting text**

Maak tekst vet dmv de<strong> tag. Dit was voorheen de <b> tag maar die is inmiddels afgeschreven en niet compatibel met HTML5.

Maak tekst cursief dmv de <em> tag (em = emphasis). Voorheen was dit de <i> tag.

Onderlijn tekst dmv de <u> tag.

Tekst kan worden doorgehaald mbv de <strike> tag.

**Les 18 – Unordered lists**

Een ongeordende lijst start met de <ul> tag. Daarbinnen kunnen list items met de <li> tag gedefinieerd worden:

<h2>Mijn ongeordende lijst</h2>

<ul>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3</li>

</ul>

</h2>

De elementen worden achter een bullet geplaatst.

**Les 20 – Ordered lists**

Zelfde als ongeordende lijst maar nu de <ol> tag gebruiken:

<h2>Mijn geordende lijst</h2>

<ol>

<lo>List item 1</lo>

<lo>List item 2</lo>

<lo>List item 3</lo>

</ol>

**Les 22 – Images**

Een foto wordt met de <img> tag toegevoegd, ook dit is een self closing tag:

<img src=”images/foto.jpg” />

In dit geval staat foto.jpg in de images folder, relatief gezien tov. de index.html file die we aan het bewerken zijn.

Je kunt ook een hyperlink gebruiken:

<img src="https://img.iex.nl/charts/IssueIntraday.ashx?sc=18&Id=350112197" />

De locatie van de foto hiervoor downloaden vanuit de browser.

De afmetingen van de foto kunnen ingesteld worden door voor de closing ‘/>’ de breedte op te geven:

width=”100”

De breedte is in pixels en moet tussen dubbele quotes staan. De browser schaalt de foto proportioneel. De hoogte kan ook ingesteld worden:

height=”200”

Door zowel hoogte als breedte in te stellen wordt de foto ingedrukt/uitgerekt.

**Les 24 – Forms**

Forms worden gebruikt voor user input op een web pagina. Voor formulieren is een server side taal zoals PHP of javascript nodig om er wat mee te kunnen doen. Deze les gaat in op het invoegen van formulieren in de webpagina, later gaan we er iets mee doen.

Het eenvoudigste formulier is een tekst inputformulier:

<form>

<input type=”text” value=”Default value”>

</form>

Dit geeft een input veld met de tekst “Default value” er in. Deze moet echter weggehaald worden. Handiger is om met een placeholder te werken:

<input type=”text” placeholder=”Enter text here…”>

Wordt nu een letter getypt dan verdwijnt deze tekst.

Een andere mogelijkheid is het invoeren van een emailadres:

- <input type=”email” placeholder=”Enter email here…”>

hiermee kan automatisch het ingevoerde adres geverifieerd worden.

Een invoer textbox:

<textarea>Optionele tekst</textarea>

De optionele tekst verschijnt in de textbox en moet manueel verwijderd worden.

Een list box:

<select>

<option>Select an option…</option>

<option>Optie 1</option>

<option>Optie 2</option>

<option>Optie 3</option>

</select>

De eerste optie (hier ‘select an option’) is default zichtbaar.

Radio buttons:

<input type=”radio” name=”color” /> Red

<input type=”radio” name=”color” /> Green

Omdat het name attribuut hier dezelfde waarde heeft kan maar één van beide buttons tegelijk aangevinkt zijn. Is de naam verschillend of ontbreekt deze dan kunnen beide aangevinkt worden.

Checkbox:

<input type=”checkbox” /> Dog

<input type=”checkbox” /> Cat

<input type=”checkbox” /> Fish

Een button:

<input type=”submit” value=”Click me!” />

Wordt de value weggelaten dan wordt default ‘submit’ in de button getoond.

Met de submit button worden de ingevoerde gegevens verzonden, dit is terug te zien in de url balk, waar in de url de variabelen met hun waarde opgenomen zijn . Dit geldt alleen voor de invoervelden die een naam hebben.

Tot zover de diverse input mogelijkheden. Voor de verwerking van de ingevoerde gegevens is PHP en/of javascript nodig. Dit komt verderop in de cursus aan bod.

**Les 26 – Links**

Een externe yperlink op de pagina:

<a href=”<http://www.google.com>”>Google link</a>

Een interne link:

<a href=”helloworld.html”>My Hello World</a>

Een interne link binnen dezelfde pagina bestaat uit 2 delen. Eerst een zg. label maken op de plek waar je naar toe wilt:

<a name=“top” />

En dan onder aan de pagina de interne link plaatsen:

<a href=”#top”>Terug naar boven</a>

**Les 28 – Tables**

Een tabel bestaat uit een header row en een aantal data rows . Header en rows hebben elk een aantal cellen. Een row wordt met de <tr> tag aangegeven, een header cell met <th> en een data cell met <td>. Een tabel ziet er dan als volgt uit:

<table>

<tr><th>Naam</th><th>Leeftijd</th><th>Geslacht</th>

<tr><td>Frank</td><td>51</td><td>Man</td>

<tr><td>Anita</td><td>47</td><td>Vrouw</td>

<tr><td>Marcella</td><td>18</td><td> Vrouw </td>

<tr><td>Sophia</td><td>14</td><td> Vrouw </td>

</table>

Hier zit al wat default styling in, de header wordt vet afgedrukt en de header en data wordt zodanig uitgelijnd dat duidelijk is wat waarbij hoort.

**Les 30 – HTML entities**

Het afdrukken van speciale tekst zoals ‘<a>’ dat niet als een link geïnterpreteerd mag worden en symbolen zoals ‘é’ en ‘ü’ zijn HTML entiteiten gedefinieerd.

In w3schools.com is een overzicht te vinden van HTML entiteiten. Deze begint altijd met een ‘&’ en eindigt met een ‘;’. Voorbeeld:

&lt;

voor het < teken en

&nbsp;

voor een non-breaking space, een harde spatie

**Les 31 – IFrames**

Met een iFrame kan content van een andere bron in de pagina ingebed worden. Bijv. een YouTube filmpje:

<iframe width="560" height="315" src="https://www.youtube.com/embed/sVC6Yo9Z8CI" frameborder="0" allowfullscreen></iframe>

Dit is rechtstreeks van YouTube gehaald door op het bewuste filmpje te klikken en ‘Delen’ en ‘Insluiten’ te selecteren. Let wel op dat voor externe links deze met ‘http://’ beginnen.

**Les 33 – Putting it all together: HTML Project**

In deze les wordt een soort website gebouwd waarin vrijwel alles wat in de voorgaande lessen is behandeld terugkomt. Rob gebruikt de site sites.codeschool.org.uk als ontwikkeltool, hiermee kan nl code ingeklopt worden en op een andere tab direct uitgevoerd worden. Maar het kan natuurlijk net zo goed met TextWrangler en een browser zoals we de hele tijd gedaan hebben.

De site die wordt getoond is een simpele eigen gebouwde site die in deze les wordt nagebouwd. Het begint met een <h1> en <h2>. Daarna een link naar het Aarde plaatje (wordt als externe link opgenomen). Het filmpje rechts hiervan komt later aan bod.

Dan een nieuwe paragraaf <p> met letterlijk de tekst overgenomen gevolgd door een horizontale lijn.

Dan een <h3> About the earth gevolgd door een unordered list <ul>. Het 3e item daarvan bevat een hyperlink die in het listitem opgenomen moet worden.

Dan weer een horizontale regel gevolgd door een <h3> Top3 earth facts. Daarna ene ordered <ol> list met 3 items.

Daarna een horizontale regel, een <h3> Earth’s nearest neighbours en een tabel. Het graden symbool opzoeken (&deg;C)

Dan weer een horizontale regel en een submit button. Deze stopt Rob in een <form>. Wat nog niet eerder is behandeld is dat deze form als target meekrijgt de site waarnaartoe wordt gesprongen als op de knop wordt gedrukt:

<form action=”http://space-facts.com/earth/”>

</form>

In eerste instantie gebruikt Rob ‘target’ ‘action’ maar dat was het foute attribuut. Met ‘target’ geef je aan waar de pagina geopend moet worden (bijv. een nieuwe tab). Met ‘action’ wordt de actie – hier de url – aangegeven als op de button wordt gedrukt.

Tot slot het iFrame met het filmpje Spinning earth van YouTube invoegen. In eerste instantie zet Rob het filmpje boven aan de pagina, maar dan lijnt het niet uit zoals in het voorbeeld waar het rechts van het plaatje staat. Hiertoe moet in het iFrame het ‘align’ attribuut toegevoegd worden:

<iframe align=”right” width=”560” height=”315” src={url van filmpje} frameborder=”0” allowfullscreen></iframe>

Let overigens op dat de YouTube link niet start met ‘http:/’ maar de link werkt prima.

**Deel 2 CSS**

**Les 38 – CSS**

CSS = Cascading Style Sheets, betreft de opmaak van een website waar HTML de inhoud betreft.

Rob gebruikt w3schools.com/css/demo\_default om aan te geven wat je met CSS kunt bereiken. Helaas is de demo veranderd: de CSS code van de demo kan niet meer getoond worden.

**Les 39 – Inline CSS**

Creëer een nieuwe folder ‘2 – CSS’ in de Web Development folder en creëer een nieuwe index.html. Haal de initiële code van example.com. Haal de opmaak code (<style> code) er uit, deze gaan we van scratch opnieuw maken.

In deze les wordt inline CSS toegepast om snel resultaat te laten zien, dit is niet de geëigende manier om style code toe te voegen, het wordt zo een ongeorganiseerde bende. De styling van elke paragraph zou apart gedefinieerd moeten worden.

Voeg het volgende aan de paragraph <p> tag toe:

<p style="color:green; font-size:150%"

**Les 41 – Internal CSS**

De geëigende manier om style info toe te voegen is internal CSS. Hier wordt in de <head> sectie styling info gegeven voor de body, voor de <div>, voor hyperlinks <a>, voor paragraphs <p>, etc etc. Zie als voorbeeld de originele HTML/CSS code van example.com.

Maak de <body> van de pagina leeg. Voeg een <style> tag toe met een definitie voor de paragraph <p>:

<style>

p {

color: green;

}

</style>

Voeg in de body 2 paragrafen met minimale tekst toe:

<p>Dit is paragraaf 1</p>

<p>En dit is nog een paragraaf</p>

Beide paragrafen bevatten nu groene tekst.

**Les 43 – Classes and ID’s**

Om paragrafen toch een eigen styling te geven biedt CSS 2 mogelijkheden: classes en ID’s.

Een class definieert styling info die vanuit alle mogelijke tags kan worden gebruikt door naar de klasse te verwijzen. Verwijder de styling info van de paragraph <p> en voeg het volgende toe:

.large {

font-size: 300%;

color: green;

}

Dit definieert de klasse die we ‘large’ noemen. Pas de eerste paragraph <p> als volgt aan:

<p class=”large”>Blablabla</p>

De tweede paragraph blijft ongewijzigd.

ID’s zijn feitelijk hetzelfde als classes maar de conventie is dat classes meerdere keren gebruikt worden en ID’s maar één keer. Voeg in de body het volgende toe:

#orange {

color: orange;

}

LET OP: de definitie van een id begint met een #.

En verander de tweede paragraph:

<p id=”orange”>blablabla</p>

Pas de tweede paragraph nogmaals aan:

<p id=”orange” class=”large”>blablabla</p>

De tekstgrootte wordt nu aangepast conform klasse ‘large’, en de kleur wordt oranje. Kennelijk heeft de id voorrang boven de klasse. Het maakt niet uit of eerst de klasse en daarna de id wordt opgenomen of omgekeerd.

Er kan binnen een tag naar meerdere klassen verwezen worden. Definieer de volgende klassen in de <head> tag:

.underline { text-decoration: underline; }

.bold { font-weight: bold; }

En pas de tweede paragraph aan:

<p id=”orange” class=”large underline bold”>blablabla</p>

De klassen worden enkel door een spatie gescheiden in de code.

Om binnen een paragraph een bepaalde klasse te gebruiken is de ‘span’ bedacht. Hiermee kan het stukje tekst tussen <span> en </span> een eigen styling gegeven worden door naar de juiste klasse te verwijzen. Voeg een derde paragraph toe:

<p>The third <span class=”large underline bold”>word</span> in this paragraph is underlined, large and bold.</p>

**Les 45 – Divs**

Een division is een logisch afgescheiden deel van de webpagina, zo kan deze bijv. bestaan uit een header, het middengedeelte en een footer. Deze kunnen op zich ook weer in divisions onderverdeeld worden.

Zet de eerste paragraph in een division (<div> en </div> eromheen). En definieer een style voor de divisions in de header:

div {

background-color:blue;

}

Nu wordt duidelijk dat een division een rechthoekig gebied is. Pas de div style verder aan:

div {

background-color:blue;

width: 100px;

}

De breedte wordt nu op 100 pixels gezet en de tekst in de division wordt automatisch gewrapped, de hoogte wordt automatisch uitgerekend zdd. de tekst er in principe in past. Het laatste wordt past echter niet en valt er deels buiten.

De hoogte kan ook manueel ingesteld worden:

div {

background-color:blue;

width: 200px;

height: 200px;

}

Vervolgens zet Rob een tweede div om de tweede regel heen en definieert aparte classes voor beide divs. In de style sectie:

.bluebox {

background-color:blue;

width: 200px;

height: 200px;

}

En voor de eerste division:

<div class=”bluebox”>

Voor de tweede division definieert Rob een aparte klasse ‘redbox’ en stopt deze in de tweede division. Niets nieuws onder de zon hier.

**Les 47 – Colors**

Tot nu toe zijn standaard kleuren gebruikt zoals ‘red’, ‘blue’, ‘green’ etc. Er kan ook met niet standaard kleuren gewerkt worden, dit gaat obv. HTML color codes. Rob verwijst naar de site html-color-codes.info waar een kleurenkaart met bijbehorende codes te zien is. De verwijzing naar een color code gaat als volgt:

.bluebox {

background-color:#DCB519;

width: 200px;

height: 200px;

}

**Les 48 – Floating**

Met het float attribuut van een style kan content links of rechts op de webpagina geplaatst worden, ipv onder elkaar zoals we tot nu toe gezien hebben. Voeg het attribuut toe aan de div style:

float: left;

Alle content met div style opmaak komt nu rechts van de andere content. Stel er is nog een andere style die rechts uitgelijnd is, dan zal de content die niet ‘floated’ is tussen de links en rechts geplaatste content worden geplaatst. Dit is goed zichtbaar als het browser window wordt uitgerekt en weer ingedrukt.

Indien de andere content onder de links en rechts geplaatste content moet komen dan daarvoor een nieuwe class met clear attribuut maken. Zet de waarde ervan op ‘both’ zodat de non-floating content zowel onder de links als rechts geplaatste content wordt geplaatst.

**Les 50 – Positioning**

Tot nu toe had alle content een standaard positie op de pagina die door de browser werd bepaald. Het is mogelijk om dit zelf te regelen. Verwijder de floating opmaak van de vorige les, zodat alles weer bij het oude is.

De purplebox gaan we 100 pixels naar rechts en 200 pixels naar beneden verplaatsen tov zijn default positie:

.purplebox {

background-color:#B40$AE;

width:200px;

height:200px;

position:relative;

left:100px;

top:200px;

}

De positie is relatief tov de default, en we stellen een linker margin tov die positie in van 100 pixels en een top margin van 200 pixels. Merk op dat de purplebox nu over de greenbox en de derde regel tekst heen staat. Dit kan aangepast worden door een style een ‘z-index’ te geven. De content met de hoogste z-index staat op de voorgrond.

Geef de purplebox een z-index van 1:

z-index:1;

En de greenbox een z-index van 2. Het resultaat is echter hetzelfde, de purplebox staat nog steeds op de voorgrond. Dit komt omdat de z-index alleen werkt als een style position data heeft. Voeg deze voor de greenbox toe:

position:relative;

Dit is voldoende, de positie van de greenbox is gelijk aan zijn standaard positie.

Wanneer de left en top margin negatief wordt genomen dan schuift de content van de pagina af. Deze kan ook niet meer zichtbaar gemaakt worden door omhoog en naar links te scrollen, het is simpelweg niet zichtbaar.

Andere attribuutwaarden voor position:

* **absolute**: een vaste plek op de pagina, de z-index werkt dan echter niet meer, de purplebox komt voor de andere content
* **fixed**: een vaste plek op het beeldscherm. Bij naar beneden scrollen blijft de content op dezelfde plek staan, wat bij headers handig is maar bij advertenties irritant.
* **static**: dit is de default

**Les 52 – Margins**

In de vorige les was sprake van overlappende content. Dit kan worden voorkomen door content een margin mee te geven van een x aantal pixels die aan alle kanten van de content wordt toegepast. De overige content wordt dan op de margin afstand gepositioneerd.

Als voorbeeld de purplebox:

margin:50px;

Dit zorgt ervoor dat de greenbox wat naar beneden schuift. In dit voorbeeld zijn de marges aan alle zijden 50 pixels. Het is ook mogelijk om de marges per zijde op te geven, in de kloksgewijze volgorde top-right-bottom-left:

margin:50px 100px 30px 10px

Voor elke zijde is ook een afzonderlijk marge attribuut beschikbaar, zodat expliciet één marge ingesteld wordt en de overige hun default waarden houden, bijv:

margin-bottom: 40px;

**Les 54 – Padding**

Padding betreft het aanbrengen van margins **binnen** een division, daar waar de margins uit de vorige les 52 betrekking hadden io de ruimte **buiten** de division.

Stel een padding van 10 pixels in voor de purplebox:

padding:10px;

Wat opvalt is dat de linker margin 10 pixels is, maar de top margin veel groter. Dit komt omdat de <p> tag waarin de tekst is opgenomen zelf een bepaalde margin bezit. Het merkwaardige is dat je dit pas ziet zodra je padding hebt toegevoegd. De greenbox heeft nl. geen padding, de tekst staat wel in een <p> tag maar er is geen bovenruimte voor de tekst binnen de container.

Merk ook op dat de padding de container groter maakt. De purplebox heeft een opgegeven breedte en hoogte van 200 pixels, maar door de padding van 10 pixels worden beide dus 220 pixels breed. Dit kan nogal eens problemen geven.

**Les 56 – Borders**

De les begint met een opgave: creëer 3 divs die ieder 100x100 pixels groot zijn, naast elkaar staan en resp. een rode, blauwe en groene achtergrond hebben.

Voor de divisions maakt Rob één style voor de afmetingen:

div {

width:100px;

height:100px;

float: left; /\* hierdoor komen ze naast ipv onder elkaar te staan

margin: 10px; /\* optioneel: een beetje tussenruimte oogt beter

}

En voor iedere kleur definieert Rob een aparte style class. Omdat deze ieder maar één keer gebruikt worden had hij net zo goed met id’s kunnen werken.

.red { /\* idem voor blue en green

background-color:red;

}

De divisions:

<div class=”red”>Division 1</div>

<div class=”blue”>Division 2</div>

<div class=”green”>Division 3</div>

Voeg aan de rode division een border toe. Om een border zichtbaar te laten zijn, moeten de dikte, kleur en type opgegeven worden:

border:5px black solid;

Merk op dat de border bij de division wordt opgeteld: deze wordt nu 110 pixels breed en hoog. Rob experimenteert nog wat met border types (zie w3schools.com) dashed, ridge, inset, dotted en outset.

Voor de blauwe division gaat Rob voor elke zijde een verschillende border creëren:

border-style:ridge solid dotted dashed; /\* top right bottom left

Omdat dikte en kleur niet zijn opgegeven worden hiervoor default waarden (1 pixel en zwart) gebruikt. De dikte en kleur kunnen ook per zijde opgegeven worden:

border-width: 10px 5px 3px 1px;

border-color: green yellow blue black;

De groene division afgeronde krijgt afgeronde hoeken:

border-radius:10px;

Hiermee kan een cirkel eenvoudig gecreëerd worden. Neem daarvoor de helft van de breedte (50px in dit geval).

**Les 58 – Fonts**

Het font kan worden aangepast met het font-family attribuut. Definieer hiervoor een style voor de paragraph:

p {

font-family: "Lucida Sans Unicode", "Lucida Grande",sans-serif

}

In principe wordt het Lucida Sans Unicode font geladen. Mocht de browser die de webpagina laadt dat niet ondersteunen, dan zal deze het meer algemene Lucida Grande font gebruiken. Mocht dat ook niet lukken dan zal de browser terugvallen op diens default sans-serif font.

Op w3schools.com staat de nodige uitleg over serif, sans serif en monospace fonts.

**Les 60 – Styling text**

Verschillende attributen om tekst op te maken:

font-weight:bold; /\* tekst vet tonen

color: green;

font-size:200%; /\* 2x zo groot als standaard

font-size:20px; /\* absolute grootte van 20 pixels

/\* NB default grootte is 16 pixels

font-size:2.4em; /\* 2.4x de standaard grootte = 240%

font-style:italic; /\* tekst schuin tonen

tekst-decoration:underline; /\* tekst onderstrepen

NB font-size in pixels geeft bij het resizen bij sommige browsers problemen (bij Chrome werkt dit prima overigens). Dat is de reden om te kiezen voor relatieve grootte in % of em.

**Les 62 – Aligning text**

Tekst is standaard links uitgelijnd. Met het tekst-align attribuut kan tekst anders uitgelijnd worden:

text-align:right; /\* tekst rechts uitlijnen

In het voorbeeld wordt tevens het erven van eigenschappen behandeld. Neem als voorbeeld de volgende styling definities:

p {

font-family: Arial, Helvetica, sans-serif;

}

.greenbox {

width:300px;

height:50px;

text-align:right; /\* justify is een andere optie

background-color:green;

}

En in de body:

<div class=”greenbox”>

<p>This is some text</p>

</div>

<p>This is some other text</p>

Omdat de eerste paragraph in de division is opgenomen neemt de <p> de attributen van de <div> over, wat betekent dat de tekst van de eerste <p> in de groene box rechts wordt uitgelijnd. De tekst in de tweede <p> erft de eigenschappen van de <div> echter niet en wordt in Arial font getoond.

NB Justify aignment geeft zowel links als rechts een kantlijn (hier in de groene box) waartegen de tekst uitgelijnd wordt.

**Les 63 – Stying links**

Een standaard hyperlink toevoegen:

<a href=<http://www.google.com>>Google</a>

Wil je de link in een nieuwe tab openen:

<a href=<http://www.google.com> target=\_blank>Google</a>

Wanneer de betreffende pagina al is bezocht dan geeft de browser deze standaard een afwijkende kleur, bijv. purple ipv blue. Deze kleur kun je ook instellen door de <a> tag een style mee te geven:

a:visited {

color:green;

}

Wanneer je de cursor over de link beweegt (hover) dan kun je de style daarvan ook aanpassen, bijv. het font van de link aanpassen:

a:hover {

font-size:32px;

}

NB Deze syntax met de “:” is nog niet eerder behandeld. CSS kent de nodige standaard ‘selectors’ - zoals visited en hover – waar style info aan gekoppeld kan worden. De site w3schools.com geeft een volledig overzicht van selectors.

**Les 66 – CSS Project: BBC News Website (1)**

De BBC News website heeft boven aan de pagina 2 rode bars die van links naar rechts lopen. De kleur bepaalt hij mbv een Chrome extension FileZilla. Safari kent dit niet maar met de app Digitale Kleurenmeter kan dit ook. Deze toont de hexcodes voor de RGB afzonderlijk zodat ze met de hand ingetikt moeten worden.

Maak eerst de index.html in TextWrangler leeg – style info en body – en save als bbc.html in de HTML dir.

Rob maakt een container <div> voor alle content, dit schijnt meer controle te geven. Voor de topbar creëert hij een ‘topbar’ id, geen class omdat er maar één topbar is. De eerste poging is als volgt:

#topbar {

width:100%; /\* volledige breedte

height:40px; /\* gegokt

background-color:770E08; /\* mbv dig kleurenmeter app

}

<div id=”container”>

<div id=”topbar”>

</div>

</div>

Er staat nu een rode balk boven aan de pagina.. Deze heeft echter een zichtbare witte margin die we niet willen. Deze is zeer waarschijnlijk afkomstig van de <body> tag. Daarvoor passen we de style aan:

body {

margin:0;

}

De BBC website heeft vele gedeelten (met tekst) die een vaste breedte hebben, die dus niet meerekken wanneer de webpagina wordt uitgerekt in de breedte. Ook de topbar heeft zo’n stuk. Hiervoor maken we een style class ‘fixedwidth’ aan:

.fixedwidth {

width:1050px; /\* na wat nameten

margin:0 auto; /\* truc om div binnen div te centreren

background-color:blue: /\* test, om effect te zien

}

Voor het fixed width stuk binnen de topbar maken we een nieuwe div binnen de topbar div:

<div class=”fixedwidth”>

BLABLA /\* test, om effect te laten zien

</div>

Het BBC logo kopieert Rob mbv het CMD-SHIFT-4 screenshot commando. Hij bewaart het in een ‘images’ folder binnen de CSS dir. Voor het logo creëert hij een div met id ‘logodiv’ binnen de fixedwidth div:

<div id=”bbclogodiv”>

<img src=”images/bbclogo.png” />

</div>

Het logo heeft wat padding nodig om los te komen van de top rand van de pagina:

#logodiv {

padding-top:5px;

}

Een nieuwe div binnen de topbar zal standaard onder de logo div geplaatst worden. Deze moeten echter rechts ervan komen, wat we bereiken door:

float:left;

aan de logodiv style toe te voegen.

Er zit ruimte tussen het bbc logo en de login icon. Dit vullen we op door een rechter border en rechter padding aan de logodiv style toe te voegen:

#logodiv {

padding-top:5px;

float:left;

border-right:1px solid #990000;

padding-right:10px;

padding-bottom:4px; /\* om ervoor te zorgen dat de border

/\* tot onderaan loopt

}

Het login logo kopieert Rob weer vanaf de BBC pagina. Hiervoor weer een aparte div maken met id logindiv maken (geen class omdat de id maar één keer wordt gebruikt):

<div id=”logindiv”>

<img src=”images/bbclogin.png” />Login

<div>

En een style:

#logindiv {

float:left;

}

De kleur van de login tekst moet wit zijn. Omdat alle tekst boven wit is, voegen we de kleur toe aan de topbar style:

color:white;

Het font is niet goed. Dit zoekt Rob op mbv de Chrome sourcecode info. Hij zet deze in de <body> style:

font-family: verdana, helvetica, arial,sans-serif;

Het font maakt hij bold en 0.7 em groot. Dat doet hij dan weer in de logindiv style:

#logindiv {

float:left;

font-weight:bold;

font-size:0.9em;

}

Vervolgens is er een gedetailleerd gerommel om login logo en tekst op de juiste plek te krijgen. Het plaatje blijkt los van de tekst omlaag bewogen te moeten worden. Daarvoor creëert Rob een aparte logindiv style voor alle images binnen de logindiv style. Omdat dat er maar één is, heeft die betrekking op het login logo. De truc is de positie relatief te maken en de top op 10 pixels (in mijn geval) te zetten waardoor het login logo 10 pixels lager staat dan zijn default positie:

#logindiv img {

position:relative;

top:10px; /\* deze waarde werkt voor mij het best

}

Daarnaast moet er nog de nodige padding toegevoegd worden om login logo + login tekst op de juiste plek te krijgen, en moet een border rechts van de logindiv geplaatst worden:

#logindiv {

float:left;

font-weight:bold;

font-size:0.9em;

padding: 6px 80px 14px 50px;

border-right:1px solid #990000;

}

Om de border rechts van de login tekst de gehele balk te laten beslaan moet er een bottom padding toegevoegd worden, in mijn geval staat die op 14 pixels.

**Les 67 – BBC News Website (2)**

De overige menu elementen laat Rob dmv een unordered list zien, omdat ze een zeer vergelijkbaar patroon hebben. Rob creëert weer een container div hiervoor. De eerste poging is als volgt:

#topmenudiv {

float: left;

}

<div id=”topmenudiv”>

<ul>

<li>News</li>

</ul>

</div>

De tekst ‘News’ wordt nu met een bullet afgebeeld. Deze halen we weg door voor de <li> binnen de topmenudiv een style aan te maken:

#topmenudiv li {

list-style:none; /\* geen bullet point oid tonen

font-weight:bold;

font-size:0.9em;

}

De ‘News’ tekst wordt te laag getoond. De vraag is waar de default padding of margin vandaan komt. Het binnen de ‘topmenudiv li’ op 0 zetten van eerst de padding en dan de margin geeft geen resultaat. Dan zou het in de default style van de <ul> kunnen zitten. Hiervoor creëert Rob een style binnen de topmenudiv style:

#topmenudiv ul {

margin:0;

}

De ‘News’ tekst wordt nu te hoog getoond, waarmee duidelijk is dat de <ul> margin moet worden aangepast . De margin-top wordt op 10- pixels gezet.

De ‘News’ tekst wordt ook teveel naar rechts afgebeeld. Dit zoekt Rob op dezelfde manier uit, het blijkt in de <ul> default padding te zitten. Het eerste resultaat is als volgt:

#topmenudiv ul {

margin:10px;

padding:0 20px 0 20px; /\* later op 0 zetten, zie hieronder

}

Padding is nog niet definitief. Als de rechter border en andere list items worden getoond wordt duidelijk wat de waarden moeten zijn. De rechter border blijkt direct naast de ‘News’ tekst te komen. Dit lossen we op door de topmenudiv <ul> padding op 0 te zetten en voor de <li> items in te stellen:

#topmenudiv li {

list-style:none; /\* geen bullet point oid tonen

font-weight:bold;

font-size:0.9em;

padding:0 20px 0 20px;

}

De rechter border begint nu te laag, dat komt door de margin en padding in de <ul> topmenudiv style. Deze zet Rob op 0 en stelt de padding in op de <li> topmenudiv style:

#topmenudiv ul {

margin:0;

padding:0;

}

#topmenudiv li {

list-style:none; /\* geen bullet point oid tonen

font-weight:bold;

font-size:0.9em;

padding:15px 20px 14px 20px; /\* voor mij de juiste waarden

}

Nu de list item style klaar is hoeven alleen de items zelf in de lijst opgenomen te worden: Sport, Weather, iPlayer, TV, Radio, More…

Het resultaat is een lange rechterborder, de list items zelf zijn niet zichtbaar. Dit komt omdat de float:left niet in de <li> topmenudiv style is opgenomen waardoor de list items onder elkaar worden geplaatst. Omdat de tekst wit is vallen ze weg tegen de achtergrond.

Tot slot moet de search bar nog toegevoegd worden. Hiervoor creëert Rob een <div>:

<div id=”searchdiv”>

<input type=”text” placeholder=”Search” />

</div>

Een id en geen class omdat we maar één searchbox hebben.

Een style voor de seardiv:

#searchdiv {

float:left;

padding:5px 0 0 10px;

}

En voor de input box een style binnen de searchdiv:

#searchdiv input {

height:25px;

padding-left:10px;

font-size:0.9em;

border:none;

}

Tot slot nog het vergrootglas in de searchbox zetten. Rob maakt een kopie van het vergrootglas op de BBC website, zet het resultaat in de images folder en stelt het in als achtergrond plaatje in de input textbox. Dit is iets nieuws vwb CSS. Voeg toe in de searchdiv input:

background-image:url(“images/vergrootglas.png”);

Dit heeft als resultaat dat de searchbox vol staat met vergrootglaasjes. Dat moet eruit en het vergrootglas moet rechts uitgelijnd worden in de box:

background-repeat:no-repeat;

background-position:right center;

Informatie hierover is te vinden op w3schools.com onder CSS Selectors.

**Les 68 – BBC News Website (3)**

In deze les wordt de balk onder de topbar gerealiseerd. Deze noemt Rob ‘newsbar’. Omdat deze qua lay-out sterk lijkt op de topbar, gebruikt Rob de topmenudiv style als basis. De body met de content maakt hij leeg. Dit blijft over:

<div id=”newsbar”>

<div class=”fixedwidth”>

</div>

</div>

Id “newsbar” is een kloon van topbar, met een andere kleur ter onderscheid:

#newsbar {

width:100%;

height:80px;

background-color:#000000; /\* andere kleur dan topbar!

color:white;

}

Rob vindt het nog nodig juist voor de newsbar div een ‘break’ div te definiëren, omdat de float nog op ‘left’ zou staan. Dat is echter niet nodig omdat alleen de lagere niveau div’s left floaten. Toevoegen van de break div haalt dan ook niets uit:

.break {

clear: both;

}

<div class=”break”>

</div>

Dan gaat hij de tekst NEWS UK toevoegen. Origineel is hiervoor een image gebruikt, Rob doet het met tekst. Mijn oplossing is als volgt:

#newsuk {

padding-top: 0px;

font-size:3.0em;

}

#uk {

padding-left:10px;

font-size:0.7em; /\* LET OP: relatief tov de 3.0em🡺2.1em

}

In de body:

<div id=”newsuk”>NEWS<span id=”uk”>UK</span></div>

Omdat de span “uk” binnen de div “newsuk” valt is de “uk” fontsize relatief tov de “newsuk” fontsize.

Vervolgens maakt Rob de RSS. Hiervoor creëert hij een div “rss”. Mijn versie:

#rss {

float:right;

margin:30px;

font-size:1.2em;

moargin-top:27px;

}

In de body:

<div id=”rss”>RSS

</div>

De RSS staat nu rechts in de balk maar onder de “newsuk” div. Als we deze float:left maken dan staan ze op dezelfde hoogte. Het RSS logo kopieert Rob van de BBC pagina. De RSS body:

<div id=”rss>RSS

<img src=”images/rsslogo.png”/>

</div>

Het plaatje is iets te groot. Hiervoor een aparte rss img style maken. Mijn oplossing:

#rss img {

height:18px;

margin-bottom:-2px;

}

Vervolgens maakt Rob het menu beneden de zojuist gemaakte news bar, ook weer dmv. een unordered list. Omdat deze binnen de “fixedwidth” div valt en onder de “newsuk” div moet komen te staan, moet er een break komen na de “rss” div:

<div class=”break”>

</div>

Het voorlopig eindresultaat is als volgt:

#topicmenudiv {

} /\* Geen styling, niet nodig op dit niveau

#topicmenudiv ul {

background-color:#3E0C0D;

padding:3px 0 3px 0; /\* nodig voor het lijntje rechts van elk item

/\* loopt niet helemaal van boven naar onder

margin:0;

height:15px;

}

#topicmenudiv li {

list-style:none;

font-size:0.8em;

border-right:1px solid #770E08;

padding:0 10px 0 10px;

float:left;

}

En de body:

<div id="topicmenudiv">

<ul>

<li>Home</li>

…overige <li> items…

<li style="border-right:none">Entertainment & Arts</li>

</ul>

</div>

Het laatste element heeft geen rechter border. Dit maken we met een inline style in orde.

Tot slot regelt Rob alvast de weergave van een geselecteerd item uit de lijst, initieel is ‘UK’ geselecteerd. Deze wordt geïnverteerd weergegeven: zwarte letters op witte achtergrond. Hiervoor creëert Rob een class:

.selected {

background-color:#EDEDED; /\* lichtgrijs

color: #666666; /\* donkergrijs

}

En in de body het UK item aanpassen dmv een CSS trucje (inline style):

<li class=”selected”>UK</li>

De kleuren zijn goed, maar het hele vakje wordt nog niet ingekleurd. Hiervoor gebruikt Rob een truc omdat de padding van de <li> de padding die we in de “selected” class willen toepassen overruled. De truc is om de <li> padding te overrulen:

.selected {

background-color:#EDEDED;

color:#666666;

height:18px;

position:relative;

top:-3px;

padding-top:3px !important;

}

Het achtergrondplaatje dat op de BBC website te zien is achter tot wat nu toe gemaakt is creëert Rob hier niet om copyright redenen. Hij geeft een tip is om de opacity in te stellen als je toch een plaatje wil laten zien.

**Deel 3 – JavaScript**

**Les 75 – What is JavaScript?**

Met JavaScript kan een website interactief gemaakt worden. HTML en CSS zijn bedoeld om de styling te regelen.

Rob geeft een voorbeeld van inline javascript:

<button onclick=”alert(‘Hi’)”>Click Me!</button>

De alert verschijnt wanneer op de button wordt geklikt. Inline javascript is niet de normale manier van javascript toepassen. Zie volgende les.

**Les 76 – Internal javaScript**

Voeg in de body een <script> tag toe en zet daarin de javaScript code:

<script type=”text/javascript”>

alert(“Page loaded!”);

</script>

Javascript kan zowel in de <head> als in de <body> worden opgenomen, zolang het maar in een <script> is opgenomen.

**Les 78 – Accessing elements**

Om elementen te kunnen benaderen vanuit javaScript moet het element een class of id hebben:

<p id=”myText”>Voorbeeldtekst</p>

<script type=”text/javascript”>

alert(“document.getElementById(“text””).innerHTML);

</script>

Hiermee wordt de tekst van de id ‘myText’ in een alert getoond.

Aanpassen van de inhoud kan ook:

document.getElementById(“text””).innerHTML = “Aangepaste tekst”;

Javascript commentaar kan in twee smaken:

* single line comment: tekst achter ‘//’
* multiline comment: tekst tussen ‘/\*’ en ‘\*/’

**Les 80 – Responding to a click**

Dit gaat in internal javascript als volgt:

<button id=”myButton”>Click Me!</button>

<script type=”text/javascript”>

document.getElementById(“myButton”).onclick=function() {

alert(“Hello!”);

}

**Les 82 – Changing website content**

Hier combineert Rob lessen 78 en 80: door op een button te klikken verandert de HTML inhoud van een <div> element. In de <body>:

<button id=”textChanger”>Change first div text</button>

<div id=”firstDiv”>This is some text</div>

<script type=”text/javascript”>

document.getElementById(“textChanger”).onclick=function() {

document.getElementById(“firstDiv”).innerHTML =

“New text for first div”;

}

</script>

Tekst toevoegen (append) kan ook:

document.getElementById(“textChanger”).onclick=function() {

document.getElementById(“firstDiv”).innerHTML =

document.getElementById(“firstDiv”).innerHTML +

“…and this is appended text”;

Als laatste vult Rob een lege <div> met HTML code:

<div id=”emptyDiv”></div>

document.getElementById(“textChanger”).onclick=function() {

document.getElementById(“emptyDiv”).innerHTML =

“<ul><li>Audi</li<li>Fiat</li></ul>”;

**Les 84 – Mini Challenge: Disappearing Circles**

Opdracht in deze les is om een rode cirkel te tonen. Wanneer er op wordt geklikt dan moet deze verdwijnen.

<style>

#redCircle {

width:200px;

height:200px;

border-radius:100px;

background-color:red;

}

</style>

In de body:

<div id=”redCircle”></div>

<script type=”text/javascript”>

document.getElementById(“redCircle”).onclick=function() {

document.getElementById(“redCircle”).style.display=”none”;

}

</script>

**Les 86 – Variables**

Voorbeeld van het aanpassen van de tekst van een <div> met door de gebruiker ingevoerde tekst:

<input id=”myInput” type=”text" value=”test” /> // default value=”test”

<button id=”textChanger”>Change the text</button>

<div id=”firstDiv”>This is the original text</div>

<script type=”text/javascript”>

var newText=getElementById(“myInput”).value;

document.getElementById(“textChanger”).onclick=function() {

document.GetElementById(“firstDiv”).innerHTML=newText;

}

Wanneer op de button wordt geklikt wordt de tekst in de div aangepast naar “test”, maar wordt daarna geklikt dan gebeurt er niets meer. Dit komt omdat variabele ‘newText’ eenmalig een waarde krijgt: de initiële waarde “test” van het myInput veld.

Om te zorgen dat de ingevoerde tekst wordt weergegeven in de div moet newText na elke klik op de button de actuele waarde krijgen van myInput:

document.getElementById(“textChanger”).onclick=function() {

newText=getElementById(“myInput”).value;

document.GetElementById(“firstDiv”).innerHTML=newText;

}

**Les 88 – Arrays**

Definieer als volgt:

var myArray = new Array();

myArray[0]=”pizza”;

myArray[1]=”chocolate”;

myArray[2]=23;

myArray[3]=-4;

Een array heeft geen vast type, elk element kan een ander type bevatten zoals hierboven. De array kan eenvoudiger gedefinieerd worden:

var myArray=[“pizza”, “chocolate”, 23, -4]; // var niet nodig????

// zie cursus video

Het weergeven van de array waarden is lastig in code, maar kan wel eenvoudig met de console log die elke browser heeft:

console.log(myArray);

Dan behandelt Rob nog een aantal methoden die op arrays toegepast kunnen worden:

* **push**(newElement): voegt newElement als laatste element toe aan de array.

Bijv. myArray.push(“coffee”) voegt “coffee” op index 4 aan myArray

* **splice**(i, n): verwijdert n elementen vanaf index i, de overige elementen schuiven op.

Bijv. myArray.splice(1,2) verwijdert de indexen 1 en 2 en zet index 3 op index 1, myArray is dan gelijk aan [“pizza”, -4].

* Met **splice** kunnen ook elementen ingevoegd worden.

Bijv. myArray.splice(2,0,”koffie”, 51) voegt “koffie” op index 2, en 51 op index 3 toe. De overige elementen schuiven op, myArray is dan [“pizza”, “chocolate”, “koffie”, 51, 23, -4]. Let op dat de tweede parameter van splice hiervoor 0 moet zijn.

* Met **length** kan het aantal elementen opgevraagd worden.

Bijv. myArray.length geeft als uitkomst 4.

**Les 90 – If statements**

Voorbeeld:

x = 1;

if (x == 1) {

alert(“x equals 1”);

} else {

alert(“x does not equal 1”);

}

De conditie moet tussen haakjes staan. Let ook op het dubbele =-teken.

**Les 92 – Javascript Game: How Many Fingers?**

Met het Math object kunnen wiskundige berekeningen gemaakt worden, bijv:

Math.pow(5,2);

Geeft 5 kwadraat.

Random getallen kunnen met de random functie gegenereerd worden:

Math.random();

Geeft een random getal tussen 0 en 1. Een getal tussen 0 en 5 (beide inclusief) krijg je als volgt:

var x = Math.random() \* 6;

x = Math.floor(x);

Hiermee heb je dus alle 6 mogelijkheden die bij het opsteken van 0 t/m 5 vingers passen.

Het spelletje wordt nu als volgt gebouwd:

<p>How many fingers am I holding up?</p>

<input id=”answer” >/

<button id=”myButton”>submit</button>

<script type=”tekst/javascript”>

var x = Math.floor(6 \* Math.random());

if (x == document.getElementById(“answer”).value) {

alert(“That’s correct!”);

} else {

alert(“That’s wrong! My number was “ + x);

}

</script>

**Les 94 – For loops**

De syntax is als volgt:

for (var i=1; i<=5; i++) {

// statements

}

Ander voorbeeld:

var myArray = [“pizza”, “chocolate”, “crisps”];

for (var i=0; i<myArray.length; i++) {

alert(myArray[i]);

}

**Les 96 – While loops**

Syntax:

var myArray = [“pizza”, “chocolate”, “crisps”];

var i=0;

while (i<myArray.length) {

alert(myArray[i]);

i++;

}

**Les 98 – Functions**

Syntax:

function add(a, b) {

return (a+b);

}

var x = add(3, 54);

**Les 99 – Javascript Project – Reaction Tester (1)**

In deze en de volgende 2 lessen 100 en 101 bouwt Rob een reactietester. Er wordt op een scherm op een willekeurige (random) plek een cirkel en vierkant getoond die moet worden aangeklikt. De tijd tussen verschijnen en aanklikken wordt vervolgens getoond.

Rob start met een rood vierkant:

#box {

width:200px;

height:200px;

background-color:red;

}

<div id=”box”></div>

Dan een javascript functie die het vierkant doet verdwijnen als er op wordt geklikt:

document.getElementById(“box”).onclick=function() {

this.style.display=”none”;

}

Omdat hetzelfde element moet verdwijnen als waar op is geklikt kan ‘**this’** gebruikt worden binnen de ‘onclick’ functie.

De box moet na een random tijd (max 5 sec) pas verschijnen, waarna er op geklikt kan worden. Hiervoor is de setTimeout functie beschikbaar. Voorbeeld:

setTimeout(function() {

alert(“done”);

}, 3000);

Deze functie heeft 2 parameters: een functie die wordt uitgevoerd als de timeout is afgelopen en de tweede parameter de timeout-tijd in ms.

In eerste aanzet laat Rob het rode vierkant na 3 seconden verschijnen, initieel is hij dus onzichtbaar:

#box {

width:200px;

height:200px;

background-color:red;

**display:none:**

}

setTimeout(function() {

document.getElementById(“box”).style.display=”block”;

}, 3000);

NB ‘block’ is de standaad display style.

De tijd moet echter niet vast maar random zijn:

var timeOut=Math.random() \* 5000;

Gebruik timeOut als parameter in setTimeout ipv 3000.

Dit werkt nu één keer, het moet echter herhaald worden. Rob maakt hiervoor de functie makeBox() die de rode box toont en de time start:

function makeBox() {

var timeOut=Math.random() \* 5000;

setTimeout(function() {

document.getElementById(“box”).style.display=”block”;

}, timeOut);

}

makeBox();

document.getElementById(“box”).onclick=function() {

this.style.display=”none”;

makeBox();

}

Vergeet de eerste makeBox() call niet, anders gebeurt er niets.

**Les 100 – Javascript Project – Reaction Tester (2)**

In deze les wordt de reactietijd bepaald: de tijd tussen verschijnen van de box en aanklikken door de speler. Hiervoor gebruikt Rob de **Date.now**() functie, waarmee de datum en tijd kunnen worden opgevraagd. Deze wordt uitgedrukt in milliseconden sinds epoque (?), ws 1-1-1980.

We houden twee tijden bij:

* createdTime: tijdstip waarop de box wordt getoond
* clickedTime: tijdstip waarop op de box wordt geklikt

Het verschil tussen beide is de reactietijd. Deze kan met een alert() worden getoond, maar Rob maakt hiervoor een <p> paragraph aan:

<p>Your time: <span id=”reactionTime”></span> seconds</p>

In de onclick functie parameter:

reactionTime = (clickedTIme – createdTime) / 1000;

document.getElementById(“creationTime”).innerHTML = reactionTime;

**Les 101 – Javascript Project – Reaction Tester (3)**

In deze les wordt de challenge afgerond. De box wordt op een random positie op het scherm getoond en kan ook van kleur veranderen. Rob start met het tonen van een willekeurige kleur. Hiervoor haalt hij code op van Stackoverflow.com en wel de functie getRandomColor die een random kleurencode van 6 hex-digits genereert:

function getRandomColor() {

var letters = ‘0123456789ABCDEF’.split();

var color = ‘#’;

for (var i=0; i<6; i++) {

color += letters[Math.round(Math.random() \* 15)];

}

}

De split() functie splitst een string op in delen die gescheiden worden door het splits-argument. In dit geval is dit een empty string, wat betekent dat elk karakter een aparte index in de array ‘letters’ krijgt.

De kleur van de box wordt als volgt in javascript aangepast:

document.getElementById(“box”).style.backgroundColor = getRandomColor();

Vervolgens wordt random bepaald of een vierkant of een cirkel wordt getoond:

if (Math.random() > 0.5) {

document.getElementById(“box”).style.borderRadius = “100px”;

} else {

document.getElementById(“box”).style.borderRadius = “0px”;

}

Tot slot moet de box op een random plek op het scherm worden getoond. Mijn oplossing hiervoor is als volgt. Allereerst moet het **position** attribute van het ‘box’ id op absolute gezet worden:

#box {

width:200px;

height:200px;

background-color:red;

display:none:

**position:absolute;**

}

Daarna wordt in de makeBox() functie de (left, top) positie random bepaald adhv de schermafmetingen:

document.getElementById(“box”).style.left =

Math.round(Math.random() \* (window.innerWidth – 200)) + “px”;

document.getElementById(“box”).style.top =

Math.round(Math.random() \* (window.innerHeight – 200)) + “px”;

Rob doet het veel simpeler met statische max waarden voor de hoogte en breedte van de beschikbare schermruimte. Hij gebruikt overigens een **relative** position.

Rob maakt het af door het uiterlijk wat attractiever te maken:

body {

Font-family: Verdana, Geneva, sans-serif; // haal op van w3schools

}

<h1>Test je reactievermogen</h1>

<p>Klik zo snel mogelijk op het vierkant of cirkel</p>

**Deel 4 – JQuery**

**Les 108 –What is JQuery?**

JQuery is een web development framework op basis van JavaScript, oftewel een JavaScript library van routines waarmee het gemakkelijker is om websites te ontwikkelen dan met de basisuitrusting HTML, CSS en JavaScript. JQuery kan gratis worden gedownload van <jquery.com>.

Er is ook een JQuery UI waarmee vanuit een grafische user interface een website kan worden ontwikkeld en tamelijk eenvoudig interactiviteit kan worden toegevoegd.

Verder zijn vele plug-ins beschikbaar voor JQuery. Een klein (gratis) aantal worden genoemd:

* <Usehook.com>: Pull-to-refresh kan hiermee worden geïmplementeerd zoals op de iPhone.
* [Tooltipster.com](Usehook.com): Hiermee kan hover functionaliteit worden ingebouwd met nog extra mogelijkheden binnen de pop-up die verschijnt.
* [Magnific Popup](dimsemenov.com): hiermee kunnen pop-ups worden getoond, zoals het uitvergroten van foto’s.
* [Unslider](unslider.com): Hiermee kunnen sliders worden geïmplementeerd.
* <Typeahead.js>: Dit is een Twitter plug-in waarmee in een input box een tekstvoorspeller wordt ingebouwd, zoals in Google Search.

**Les 109 – Embedding JQuery**

Om JQuery te gebruiken moet het in de webpages worden ingebed. Dat gaat als volgt.

Creëer een directory “4. JQuery” binnen de “Web Development” folder en zet hierin de index.html van de “3. JavaScript” folder. Haal de style en body eruit en noem de title “Learning JQuery”. Voeg de regel

<script type=”text/javascript” src=””></script>

toe aan de <head> sectie. De src wordt straks ingevuld. JQuery kan op twee manieren worden toegevoegd.

De ene manier is om het te downloaden van de jquery website. Kies naar wens versie 1 of 2 en selecteer de gecomprimeerde versie. Wanneer de betreffende pagina is geladen sla deze op als “jquery.min.js” in de “4. JQuery” folder.

LET OP: In Safari moet bij opslag het type ‘als bronpagina’ gekozen worden, niet ‘webpagina!

Vul bij de src eveneens “jquery.min.js” in. Een andere filenaam is uiteraard ook mogelijk.

De andere manier is om via een link te verwijzen naar de library. Vul bij src dan <http://code.jquery.com/jquery-latest.min.js> in.

Het voordeel van het zelf binnenhalen van JQuery (manier 1) is dat je precies weet welke JQuery versie je hebt. Het voordeel van methode 2 is dat je de laatste versie altijd hebt (wat meestal gewenst is) en dat het laden van je website waarschijnlijk wat sneller gaat omdat de bezoeker de library waarschijnlijk al eerder, vanaf een andere site, geladen heeft en dus niet meer van jouw website hoeft te laden.

Er kan op eenvoudige wijze worden vastgesteld of JQuery geladen is. Hiervoor moet de variabele jQuery bestaan:

if (typeof jQuery != “undefined”) {

alert(‘jQuery is installed!”);

} else {

alert(‘jQuery is not installed!’);

}

Dit moet op deze manier. De variabele jQuery bestaat alleen als JQuery is geïnstalleerd. Is JQuery niet geïnstalleerd dan bestaat de variabele niet en kan niet op de waarde ervan worden getest, dit zou nl. een fout in het script geven waardoor de foutmelding alert niet wordt getoond.

**Les 111 – Detecting a click**

In de komende lessen worden zaken die met Javascript zijn gemaakt nu met JQuery gebouwd. Gebruik de index.html van de vorige les, haal de jQuery detectie en de alerts er uit en creëer een groene cirkel en twee rode vierkanten. In de <head> sectie:

<style>

#green-circle{

height:200px;

width:200px;

background-color:green;

border-radius:100px;

}

.square{

height:200px;

width:200px;

background-color:red;

margin-top:10px;

}

</style>

en in de <body> sectie:

<div id=”green-circle”></div>

<div class=”square”></div>

<div class=”square”></div>

De JQuery code om te detecteren dat op de cirkel of één van de vierkanten is geklikt:

$(“#green-circle”).click(function() {

alert(“op cirkel geklikt”);

});

$(“.square”).click(function() {

alert(“op vierkant geklikt”);

});

Uit het stukje code voor de square click blijkt dat JQuery krachtiger is dan Javascript: hier worden meerdere elementen (square div’s in dit geval) mee afgevangen. Dit kan ook in Javascript maar is gecompliceerder. Het kan nog eenvoudiger door alle div’s met de volgende code af te vangen:

$(“div”).click(function() {

alert(“div geklikt”);

});

**Les 113 – Changing website content**

Gebruik de code van de vorige les. Voeg een paragraph met tekst toe boven de cirkel:

<p>Dit is een stuk tekst</p>

En verander deze als op de groene cirkel wordt geklikt:

$(“#green-circle”).click(function() {

$(“p”).html(“En nu is de tekst veranderd”);

});

Dit verandert alle <p> teksten als op de groene cirkel wordt geklikt.

De inhoud van een html element kan ook opgevraagd en getoond worden:

$(“#green-circle”).click(function() {

alert($(“p”).html());

});

Dit kunnen we ook doen wanneer over een html element gehoverd wordt:

$(“#green-circle”).hover(function() {

$(“p”).html(“Tekst is aangepast na hover”);

});

NB Let op dat hiermee worden alle <p> elementen aangepast. Wil je van één <p> element de tekst aanpassen dan moet deze een id krijgen waaraan in JQuery gerefereerd wordt.

Het is ook mogelijk een attribuut van een html element aan te passen. Dit wordt gedemonstreerd adhv een iframe waarin een webpagina is afgebeeld. Wanneer op de groene cirkel wordt gehoverd wordt de url dmv een alert getoond:

$(“#green-circle”).hover(function() {

alert($(“iframe”).attr(“src”));

});

Merk op dat de alert 2x wordt getoond. Dit gebeurt namelijk zowel bij het ‘mouseenter’ event als bij het ‘mouseleave’ event die beide de hover functie triggeren. Wil je één alert, dan één van beide functies gebruiken ipv hover:

$(“#green-circle”).mouseenter(function() {

alert($(“iframe”).attr(“src”));

});

Het aanpassen van de url gaat door de attr functie een tweede parameter met de nieuwe url mee te geven:

$(“#green-circle”).hover(function() {

$(“iframe”).attr(“src”, “http://www.iex.nl”);

});

**Les 115 – Changing styles**

Gestart wordt met het opvragen van de style attributen van de vierkanten wanneer op één van de vierkanten wordt geklikt:

$(“.square”).click(function() {

alert($(“.square”).css(“width”));

});

Vervolgens wordt het aangepast zodanig dat de breedte van het element waar op wordt geklikt wordt getoond:

$(“#div”).click(function() {

alert($(this).css(“width”));

});

De functie ‘this’ refereert naar het element waarop is geklikt.

Het aanpassen van een attribuut gaat net zoals bij de attr functie:

$(“#div”).click(function() {

$(this).css(“width”, “400px”);

});

Het verbergen van een element als er op geklikt wordt:

$(“#div”).click(function() {

$(this).css(“display”, “none”);

});

Als alternatief kan de ‘hide’ functie worden gebruikt:

$(“#div”).click(function() {

$(this).hide();

});

De ‘fadeout’ functie laat een html element geleidelijk verdwijnen:

$(“#div”).click(function() {

$(this).fadeOut();

});

De ‘fadeOut’ functie heeft veel parameters waar in de volgende les wat uitgebreider op terug wordt gekomen.

NB Let op de hoofdletter ‘O’ in fadeOut, met een kleine ‘o’ werkt het niet!

**Les 117 – Fading content**

In deze les wordt een aantal mogelijkheden van de fadeOut functie behandeld. Op jquery.com is alle informatie over deze functie te vinden.

Maak twee buttons, de ene om elementen uit te faden, de andere om ze weer zichtbaar te maken. In de body:

<button id=”fadeOut”>Fade out</button>

<button id=”fadeIn”>Fade in</button>

In de <script> sectie:

$(“#fadeOut”).click(function() {

$(“div”).fadeOut();

});

$(“#fadeOut”).click(function() {

$(“div”).fadeIn();

});

De snelheid van de fade kan als parameter worden meegegeven, de tijd is in ms (default 400 ms):

$(“#fadeOut”).click(function() {

$(“div”).fadeOut(2000);

});

Deze parameter kan ook als string meegegeven worden, de mogelijkheden zijn “fast” en “slow”:

$(“#fadeOut”).click(function() {

$(“div”).fadeOut(“slow”);

});

Het is ook mogelijk een callback functie als parameter mee te geven die wordt aangeroepen als de fade klaar is:

$(“#fadeOut”).click(function() {

$(“div”).fadeOut(2000, function() {

alert(“Fade out klaar”);

});

});

**Les 119 – Animating content**

Fading is een voorbeeld van animatie, iets waarvoor JQuery goede mogelijkheden biedt in de vorm van de ‘animate’ functie.

Wanneer op de groene cirkel wordt geklikt wordt in 1,5s de breedte, hoogte en straal verdubbeld:

$(“#green-circle”).click(function() {

$(this).animate({

width:”400px”,

height:”400px”,

borderRadius:”200px”,

marginLeft:”100px”,

marginRight:”100px”

}, 1500);

});

Let op dat de naamgeving van de attributen in Javascript anders is dan in CSS. In CSS worden kleine letters gebruikt en zijn afzonderlijke woorden in de attribuutnaam gescheiden door een streep. In Javascript wordt de ‘camel’ notatie gebruikt: de attribuutnaam start met een kleine letter en afzonderlijke woorden starten met een hoofdletter.

**Les 121 – Loading data with AJAX**

AJAX = Asynchronous JavaScript and XML

Met deze door Google ontwikkelde technologie kan een file asynchroon van de webserver geladen worden zonder de pagina te verversen, waardoor de gebruiker op de voorgrond verder kan gaan met interactie met de website. De te laden file hoeft niet perse een XML file te zijn maar kan elk type file zijn, zoals een JSON file of HTML file zoals we zullen zien. AJAX is ook niet expliciet voorbehouden aan JQuery, kan ook vanuit andere frameworks gebruikt worden.

AJAX werkt echter alleen met het http: protocol en niet met het file: protocol. Dit houdt in dat je AJAX niet lokaal kunt gebruiken maar dat het op een webserver moet draaien.

Maak op de webserver een directory “files” aan en creëer hierin een nieuwe file “1-index.html”. Zet hierin de inhoud van de index.html file die in de vorige les is gebruikt. Creëer nu in de file directory een file “test.html”. Zet hierin niet meer dan een teststring:

Dit is de inhoud van de test.html file tbv AJAX demo

Voeg nu in 1-index.html in de <script> sectie de volgende code toe:

$.get(“test.html”, function(data) {

alert(data);

});

Dit haalt de test.html file op en stopt de inhoud ervan in de variabele ‘data’.

De ‘get’ functie werkt op zich goed maar in plaats daarvan wordt de ‘ajax’ functie meer gebruikt omdat deze veel meer mogelijkheden en controle geeft. Deze kent bijv. een ‘fail’ methode voor het geval de internet connectie down is. Bij de ‘get’ methode gebeurt er dan helemaal niets. Bij de ‘ajax’ methode kun je de gebruiker hiervan via een alert op de hoogte stellen.

Vervang de ‘get’ code door:

$.ajax({

[url:”test.html](url:)”

}).done(function(data) {

alert(data);

)};

De ajax functie kent behalve de ‘url’ parameter nog veel meer parameters. De “done” methode van de ajax functie wordt aangeroepen als de data is binnengehaald.

Tot slot doet Rob nog een paar dingen met de <h1> tekst. Vervang de koptekst door de inhoud van test.html:

$(“h1”).html(data);

Of zet de inhoud van test.html achter de koptekst:

$(“h1”).append(data);

In beide gevallen komt de code ipv. de alert.

**Les 123 – Regular expressions**

Hiermee kunnen patronen van strings gezocht worden, bijv. om uit te zoeken of een ingevoerde string het format van een e-mailadres heeft. Voorbeeld:

var regex = /great/;

var string = ”Regex is great”;

result = string.match(regex);

alert(result);

Dit geeft in de popup de waarde ‘great’.

Vervang string door “Regex is hard”. Het resultaat in de popup is nu ‘null’.

Let op dat de match functie case sensitive is, “Regex is Great” geeft dus ook null terug. Wil je een case insensitive search dan moet een ‘i’ aan regex toegevoegd worden:

var regex = /great/i;

Het opzoeken van alle voorkomens van een string in een andere string kan ook, dan moet de ‘g’ (global) vlag toegevoegd worden:

var regex = /e/g;

De match functie geeft nu alle voorkomens van de letter e en de popup laat ‘e,e,e’ zien.

**Les 125 – Form validation**

In deze les wordt een volledige functionerend invulformulier gebouwd obv de opgedane kennis van HTML, CSS, JavaScript en JQuery.

Gestart wordt met de bouw van het formulier zelf. Zie hiervoor de code uit de les (teveel om hier weer te geven). Alleen de bijzonderheden worden hier vermeld.

Nieuw is de toepassing van het <label> element, dat via een id gelinkt wordt aan een <input> element:

<label for=”password”>Password</label>

<input name=”password” type=”password”/>

Wat hier nu precies de bedoeling van is wordt niet duidelijk. Echter bij buttons of checkboxes kun je ook op het label klikken alsof je op de button c.q. checkbox had geklikt.

De labels en inputs worden vervolgens in een <form> gezet zodat ze gezamenlijk behandeld kunnen worden.

Dan volgt de validatie van het formulier. Dit kan door de klik op de submit knop af te vangen, maar gebruikelijker is het om de <form> af te vangen. Het voordeel daarvan is dat de ingevoerde waarden dan ook kunnen worden benaderd. Voeg de volgende code toe in de <script> sectie:

$(“#inputForm”).submit(function(event) {

alert(“test”);

});

Wanneer op de submit knop wordt gedrukt dan wordt het formulier ‘submitted’. Op dat moment wordt de function met parameter ‘event’ uitgevoerd. De event parameter betreft het submitten van het formulier.

Wat opvalt is dat de ingevoerde waarden verdwijnen wanneer op de submit knop wordt gedrukt, dit is ook te zien in de URL balk, hierin staan de waarden van de parameters vermeld die we later in PHP met een Get functie gaan opvragen.

Het leegmaken van de invoervelden is echter niet wat we willen omdat we de ingevoerde waarden nog moeten checken voordat deze submitted worden. Vervang hiertoe de alert door:

event.preventDefault();

Nu kunnen de ingevoerde waarden worden gecontroleerd. Dit gebeurt met reguliere expressies. Voor het valideren van het email adres gebruikt Rob een functie ‘isValidEmailAddress’ met een reguliere expressie die hij van Stackoverflow.com afhaalt, dit is te veel om hier op te nemen dus zoek op in de code. De emailaddress <input> krijgt nog een id ‘email’, zodat de waarde opgevraagd kan worden. De call ziet er als volgt uit:

if (!isValidEmailAddress($(“#email”).val())) {

/\* code voor tonen foutmelding

}

De geldigheid van het email adres geef ik aan met een groen vinkje (V) of een rood kruis (X). Rob doet het wat anders. Hij creëert een div ‘error’:

<div id=”error”></div>

Deze is initieel niet zichtbaar en wordt zichtbaar gemaakt na een druk op de submit knop. Rob gebruikt deze om alle foutmeldingen weer te geven. Zie de code.

Het ingevoerde telefoonnummer wordt enkel op numeriek zijn getest (<input> phone moet nog de id ‘phone’ krijgen):

if (!$.isNumeric($(“#phone”).val())) {

/\* code voor tonen foutmelding

}

De ingevoerde passwords moeten gelijk aan elkaar zijn. Maak een id ‘pass1’ en ‘pass2’ aan voor beide velden en het if-statement wordt dan:

if $(“#pass1”).val() != $(“#pass2”).val()) && $(“pass1”.val() != “” {

/\* code voor tonen foutmelding

}

Rob vergeet te testen of er überhaupt een password is ingevuld, dat heb ik er nog bij gedaan.

Tot slot moet er wat gebeuren na het checken van de invoervelden. Rob doet dit aan de hand van de waarde van de errormessage, die hij bij elke fout-check vult met een bij het invoerveld horende foutmelding:

if (errorMessage == “”) {

alert(“all inputs ok!”);

}

**Les 127 – Introducing jQuery UI**

jQuery UI is een library die UI interacties, effecten, widgets en thema’s gebouwd op jQuery bevat en gaat zo nog een stapje verder dan jQuery zelf bij het supporten van ontwikkelaars bij het bouwen van interactieve web applicaties.

Kijk hiervoor op jqueryui.com. Om de UI te kunnen gebruiken moet jQueryUI ingebed worden in de code zoals jQuery zelf:

<script type=”text/javascript” src=[jQueryUI link]></script>

Zie voor de exacte link de video of zoek op Google (versie is inmiddels alweer hoger dan in de video).

Net als jQuery kan de UI worden gedownload of de webversie worden gebruikt, zoals hier met de door Google gehoste versie is gedaan.

Rob begint met een voorbeeld van een draggable (‘sleepbaar’) element:

In de <style> sectie:

#square {

height:200px;

width:200px;

background-color:grey;

}

In de <body>:

<div id=”square”></div>

En in de <script> sectie wordt het element draggable gemaakt:

$(“#square”).draggable();

Dit is al wat nodig is om het element op het scherm heen en weer te kunnen slepen. Er kan nog veel meer mee gedaan worden dan alleen heen en weer slepen, dit is allemaal op de jQueryUI website te vinden bij de ‘draggable’ functie.

**Les 129 – Draggables and resizables**

Hier wordt weer een draggable vierkant gebouwd, identiek aan de vorige les. Daarnaast wordt een resizable element toegevoegd. Hiervoor is een aparte ‘jQuery UI CDN” library nodig die moet worden ingebed (CDN = Content Delivery Network). Deze kan vanuit de Google library worden geïmporteerd, zoals dat met de jQuery UI library gedaan is. Zie de code voor de link.

Het resizen gebeurt met de resizable functie, op identieke wijze als de draggable functie.

**Les 132 – Droppables**

Een droppable element wordt gebruikt om een draggable element in te ‘droppen’. Wanneer dat is gebeurd kan in de code een actie uitgevoerd worden. Voeg een groter vierkant toe waar de eerste in kan worden gedropt:

In de <style> sectie:

#square2 {

height:400px;

width:400px;

background-color:yellow;

}

In de <body>:

<div id=”square2”>THIS IS A DROPPABLE SQUARE</div>

In de <script> sectie:

$(“square2”).droppable({

drop: function(ui, event) {

$(“#square2”).css(“backgroundColor”, “green”);

}

});

Dit betekent zoveel als wanneer een element in square2 gedropt is, de background-color van square2 groen wordt.

**Les 134 – Accordion & Sortables**

De accordion functie is bedoeld om van een grote hoeveelheid tekst alleen wat gewenst is te laten zien. Ter illustratie maakt Rob een flink aantal <h3> met ieder een <p> met tekst aan:

<h3>Dit is een titel</h3>

<div>

<p>Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst. Dit is een stuk tekst.</p>

</div>

Het bovenstaande een flink aantal keren herhalen.

Zet het totaal vervolgens in een div en noem deze “accordion” oid.

Voeg in de <script> sectie de accordion functie toe:

$(“#accordion”).accordion();

De tekst wordt nu in elkaar geklapt, alleen de headers zijn nog zichtbaar en als er op een header wordt geklikt wordt de bijbehorende tekst geopend en de tekst die geopend was wordt gesloten. Dit is handig voor een FAQ toepassing.

De sortable list is een handige uitbreiding op de standaard <ul> list. De elementen kunnen nu in de gewenste volgorde in de lijst worden neergezet. De voorbeeldcode:

In de body:

<ul>

<li>Pizza</li>

<li>Brood</li>

<li>Vlees</li>

<li>Pasta</li>

</ul>

In de <script> sectie:

$(“ul”).sortable();

**Les 137 – Codeplayer**

Dit is een project waarin een codeplayer ala JS.bin wordt gebouwd.

We starten met een <div> wrapper voor de hele pagina, dit is gebruikelijk. Daarbinnen wordt de topbar aangebracht in de <body> sectie:

<div id=”wrapper”>

<div id=”topbar”>

/\* Lege ruimte voor <div’s> voor straks

</div>

</div>

In de <style> sectie:

#topbar {

width:100%; /\* de gehele breedte van de pagina

height:60px;

background-color:#F1F1F1;

border-botom:1px solid grey;

}

Er verschijnen nu witte randen, die halen we weg door de margin en padding van de <body> op 0 te zetten:

body {

margin:0;

padding:0;

font-family: /\* zie link CSS-tricks.com in video

}

Vervolgens voegen we links in de topbar een logo toe. Binnen de topbar <div>:

<div id=”logo”>CodePlayer</div>

Voor het font haalt Rob op CSS-tricks.com een mooie font-family declaratie op en zet deze in de <body> style. Het logo kan nu gestyled worden:

logo {

padding: 5px 0 0 20px;

font-weight:bold;

font-size:120%;

}

Dan maakt Rob de Run button rechts op de pagina aan. Hij creëert hiervoor een <div> waarbinnen hij de <button> opneemt:

<div id=”buttonDiv”>

<button id=”runButton”>Run<button>

</div>

#buttonDiv {

float:right;

padding: 5px 10px 0 0;

}

#runButton {

font-size: 120%;

}

Waarom hij hier 2 aparte id’s voor aanmaakt is niet duidelijk, het kan nl. m.i. net zo goed in de <button> style geregeld worden. Kennelijk is het stylen op deze manier eenvoudiger te regelen?

Voor de toggle buttons in het midden start Rob met een <ul>:

<ul id=”toggles”>

<li>HTML</li>

<li>CSS</li>

<li>JS</li>

<li>Result</li>

</ul>

Als je dit runt ziet het er niet uit, de verticale list met dots staat links door het logo heen en er is witte ruimte boven de topbar ontstaan. Dit laatste verhelpen we door de <ul> margin op 0 te zetten. De dots halen we weg door de list-style op ‘none’ te zetten. De list items krijgt Rob op dezelfde regel door op <li> niveau een float left toe te voegen. De gehele lijst wordt in het midden gezet door een width van 200px en een margin ‘0 auto’ te geven.

LET OP: centreren werkt alleen als je zowel een width als een margin ‘0 auto’ opgeeft!

Het is nu nog een kwestie van de styling afronden. Stel de hoogte en border in van “toggles”. Stel de border-right van de ‘toggle li’ in op standaard ‘1px solid grey’. De streepjes staan nu achter de list items maar beginnen niet aan de rand van “toggle”. Dit lossen we op door de padding uit “toggle” te halen en deze in de ‘toggle li’ toe te voegen. Het ‘Result’ item heeft nu nog rechts een border staan die weg moet, dit lossen we dmv een inline style op:

<li style=”border:none”>Result</li>

De styling van de toggles is nu als volgt:

toggles {

width:200px;

height:27px;

margin:0 auto; /\* staat in het midden, samen met width!

position:relative; /\* zodat de toggle border

top:10px; /\* in het midden staat

padding:0; /\* zodat de <li> rechter borders de bovenkant

/\* raken

border:1px solid grey;

border-radius:3px;

}

#toggles li { /\* beinvloedt alleen ‘toggles’ li items

float:left; /\* werkt beter dan ‘display:inline’

padding:11px; /\* centreer de list items in de toggle bar en zorgt

/\* dat de streepjes boven- en onderkant ervan

/\* raken

border-right:1px solid grey;

}

Omdat er float left en right <div>’s in de topbar zitten moeten we de floats breken, (= een empty div met clear:both toevoegen) om ervoor te zorgen dat de div’s in de topbar ook op het scherm als onderdeel van de topbar zichtbaar zijn en er niet onderuit steken, hoewel Safari hier geen last van heeft:

<div style=”clear:both”></div>

Voor de code windows onder de topbar creëert Rob een class “code Container”, omdat de 4 verschillende containers op elkaar lijken. Ze krijgen wel ieder een eigen id, bijv:

<div class=”codeContainer” id=”htmlContainer”>

TEST TEKST

</div>

De eerste opzet voor de styling van codeContainer is als volgt:

.codeContainer {

float:left; /\* omdat er meerdere code containers zijn

/\* die naast elkaar komen

height:100%; /\* de volledige hoogte, echter moet de topbar

/\* hier nog van af 🡪 zie verder

position:relative; /\* tbv codeIndicator, zie verder op

width:50%; /\* eerste aanzet

background-color:blue; /\* bij wijze van test

}

Run de code en het blijkt dat er slechts één regel blauw wordt. Dit komt omdat de 100% te veel is, de topbar moet er nog af (dus 100% - hoogte topbar). Er zijn verschillende manieren in CSS om dat te doen (even googlen), Rob maakt iets in JavaScript/JQuery:

var topbarHeight = $(“#topbarHeight”).height();

var windowHeight = $(window).height();

var codeContainerHeight = windowHeight – topbarHeight;

$(“.codeContainer”).height(codeContainerHeight + “px”);

Opmerkingen:

* de JQuery height() functie werkt het meest intuïtief, deze geeft de hoogte in aantal pixels, zonder de “px” extensie. In geval van javaScript werkt “.style.height” niet (geeft leeg resultaat na lang zoeken) en moet “.clientHeight” of “.offsetHeight” gebruikt worden (google voor verschil).
* window heeft betrekking op het window object van de browser dat in beeld is
* Mbv de JQuery height() functie kan de hoogte ingesteld worden, niet vergeten “px” erachter te zetten. Een berekening in de height() functie zetten werkt niet in combinatie met de “px”, maar wel als de “px” wordt weggelaten zoals in $(“.codeContainer”).height(windowHeight – topbarHeight);

In de htmlContainer <div> wordt een <textarea> geplaatst. Dit is vergelijkbaar met een <input> maar beslaat meerdere regels. In ons geval moet de volledige ruimte worden benut:

<textarea>Start coding here…</textarea>

textarea {

width:100%;

height:100%;

border:none;

border-right:1px solid grey;

}

Het textarea beslaat nu de volledige codeContainer <div>. Om duidelijk te maken dat dit de HTML container is plaatsen we een codeIndicator rechtsboven in de codeContainer <div>:

<div class=”codeIndicator”>HTML</div> /\* NB binnen de codeContainer!

.codeIndicator {

position:absolute; /\* absolute binnen codeContainer

top:5px; /\* 5px onder de top

right:5px; /\* 5px links van de rechterkan

}

Om ervoor te zorgen dat de codeIndicator ook binnen de codeContainer wordt getekend, moet de codeContainer ‘gepositioneerd’ worden – anders heeft de ‘absolute’ waarde geen enkel effect -, wat betekent dat deze een niet-static positie moet krijgen, Rob kiest voor relative, zonder de coördinaten te veranderen.

Vervolgens worden de codeContainers aangemaakt voor het CSS en JS deel, hiervoor wordt de htmlContainer code gekopieerd. Omdat zowel cd CSS als JS container bij aanvang niet zichtbaar mogen zijn, maken we voor beide <id>’s een style aan en maken ze onzichtbaar:

#cssContainer, #jsContainer {

display:hide;

}

Voor de output maken we de resultContainer aan. Deze bevat geen <textarea> maar een <iframe>, daarnaast is deze initieel zichtbaar:

<div class=”codeContainer” id=”resultContainer”>

<div class=”codeIndicator”>Result</div>

<iframe></iframe>

</div>

Het <iframe> moet nog gestyled worden:

iframe { /\* we hebben maar 1 iframe dus uniforme styling is ok

width:99%; /\* LET OP: geen 100% anders border

/\* van codeContainer onzichtbaar

height:100%;

position:relative; /\* voorkomt overlap met htmlContainer

left:5px;

border:none; /\* border voegt niets toe

}

Tot zover de styling. Nu worden de toggles geïmplementeerd in JavaScript. Hiervoor gebruiken we een nog niet eerder gebruikte functie toggleClass() en voorzien we de toggle list items van een class (om de click af te vangen) en creëren we een class “selected” om de achtergrondkleur te veranderen:

.selected {

background-color:grey;

}

<li class=”toggleButton”>CSS</li> /\* ook voor overige list items

$(“.toggleButton”).click(function() {

$(this).toggleClass(“selected”);

});

De toggleClass functie voegt de betreffende class toe aan het HTML element of verwijderd deze, afhankelijk van of de class al dan niet bij het HTML element hoort.

De codeContainer <div>’s worden op enigszins vergelijkbare manier zichtbaar gemaakt. Rob bedient zich van een truc, door de HTML inhoud van het ingedrukte element te gebruiken als identificatie van de betreffende codeContainer:

var activeDiv = $(this).html(); /\* HTML, CSS, JS of Result

var activeContainerID = “#” + activeDiv + “Container”;

$(activeContainerID).toggle(); /\* zet element aan of uit (hide/display)

Let op dat het begin van de id-namen exact gelijk is aan de html inhoud, dus hoofdletters!

Tot slot moet de breedte van de codeContainers aangepast worden. De breedte is gelijk aan 100% / #zichtbare codeContainers:

var nVisibleCodeContainers = $(“.codeContainer”).filter(function() {

return ($(this).css(“display”) != “none”);

}).length;

De filter functie zet een filter op de verzameling codeContainer elementen. Alleen de elementen die zichtbaar zijn blijven in de verzameling, dit regelt function() in de filter() call. Met “.length” krijg je het aantal elementen uit deze verzameling terug.

NB. Function() heeft 2 parameters: index en element, maar als je ze niet gebruikt kun je ze in de aanroep weglaten. Tja….

Dit kan ook op een andere manier verkregen worden, nl. obv. de toggle buttons:

var nVisibleCodeContainers = $(“.toggleButton”).filter(“.selected”).length;

Deze filter de toggleButtons waar de selected class actief is en neemt daar vervolgens het aantal van.

De breedte per zichtbare (!) codeContainer wordt nu als volgt ingesteld:

var codeContainerWidth = 100 / nVisibleCodeContainers;

$(“.codeContainer”).css(“width”, codeContainerWidth + “%”);

Als laatste moet de in de html-, css- en javascript codeContainer ingetikte code uitgevoerd worden in de Result codeContainer als op de Run button wordt geklikt. Omdat dit nog niet eerder toegepaste javaScript nodig heeft start Rob met het vullen van het iFrame van de Result codeContainer met wat test tekst:

$(“runButton”).click(function() {

$(“iFrame”).contents().find(“html”).html(“Test tekst”);

});

Dit werkt als volgt: neem van alle iFrames de contents, dat zijn de ‘kinderen’ van het geselecteerde element, dus niet de html inhoud maar de vanuit de DOM bekeken. Zoek binnen de contents de html elementen en stel de inhoud daarvan in op “Test tekst”.

Maak nu id’s aan voor de textareas: htmlCode, cssCode en jsCode. De HTML en CSS code slaan we voor de overzichtelijkheid op in lokale variabelen:

myHTMLCode = $(“#htmlCode”).val();

myCSSCode = $(“#cssCode”).val();

De HTML en CSS code wordt uitgevoerd door deze in het iFrame op te nemen:

$(“iFrame”).contents().find(“html”).html(“<style>” + myCSSCode

+ “</style>” + myHTMLCode);

De javaScript code kan mbv de eval() functie worden uitgevoerd. Dit is een zeer riskante functie die toegang geeft tot de resources op de webserver. Gebruik ervan wordt sterk afgeraden, maar voor deze cursus die we lokaal draaien is het geen probleem.

eval($(“#jsCode”).val());

Dit werkt op zich zij het dat de code niet in het Result iFrame draait. Het iFrame geven we een id “resultFrame”. Dit benaderen we dmv javaScript code:

document.getElementById(“resultFrame”).contentWindow.

eval($(“#jsCode”).val());

De property ‘contentWindow’ geeft het window object dat is gegenereerd door het iFrame element.

**Section 5: Twitter Bootstrap**

**Les 140 – What is Bootstrap?**

Bootstrap is een javascript library gebouwd door Twitter en is bedoeld om ‘responsive design’ in websites in te bouwen. Als voorbeeld laat Rob de Bootstrap site (**getbootstrap.com**) zien. Hier staat een verhaal met plaatjes op. Wanneer je de breedte van de browser kleiner maakt, dan worden de plaatjes automatisch kleiner op de pagina. Maak je de breedte zo klein dat het niet meer past, dan gaat de site over naar zg. ‘stacked design’, wat inhoudt dat de mobiele variant van de site wordt getoond. Alles wordt dan vertikaal weergegeven, wat inhoud dat plaatjes onder elkaar ipv naast elkaar worden getoond om zo groot mogelijk weergegeven te worden in de beschikare ruimte.

**Les 141 – Bootstrap Hello World**

Net als jQuery kan Bootstrap worden gedownload en lokaal worden benaderd of kan de laatste versie via een externe link worden benaderd. Rob download Bootstrap, omdat de focus in de cursus op offline ontwikkelen ligt.

De file die wordt gedownload is een zipfile. Maak een nieuwe folder ‘5 Bootstrap’ aan, zet de file daar in en unzip de file. Er zijn 3 dir’s aangemaakt: css, fonts en js. Bij css en js kun je kiezen uit de normale versie of een minimized versie (‘min’ in de filenaam). Die is voldoende en zal wat sneller werken ivm de grootte.

Rob zet de drie folders in de root van de ‘5 Bootstrap’ folder. Dit doet hij omdat het dan correspondeert met de voorbeeld code op de getting-started pagina van de Bootstrap site.

**Les 143 – The grid system**

Wat Bootstrap feitelijk doet is de styling afhankelijk maken van de screensize. Bootstrap maakt onderscheid tussen 3 verschillende screensizes (small-medium-large) die overeenkomen met de momenteel verkrijgbare apparaten (smartphone, tablets, desktops). Per screensize klasse zijn verschillende CSS klassen ontwikkeld. De werking is gebaseerd op het toepassen van een grid op het scherm. Het grid is opgebouwd uit 12 kolommen.

Zie voor meer info de Bootstrap website. Hier staat een voorbeeld van een grid dat per regel een andere indeling heeft. Wanneer de breedte van het browser window wordt verkleind, dan worden de cellen in eerste instantie minder breed en de tekst erin wat kleiner. Verklein je het tot beneden de ‘smal’ size, dan worden de cellen ‘stacked’ (boven elkaar) weergegeven.

**Les 145 – Navbars**

Bootstrap heeft diverse templates beschikbaar waarmee snel een indruk van de mogelijkheden van Bootstrap verkregen kan worden. In deze les wordt een ‘nav bar’ gemaakt obv een template. Hiervoor gebruikt Rob als basis de Jumbotron template. De navigation bar van dit template gaat hij in deze les nabouwen.

Start met de navbar div:

<div class=”navbar navbar-default”> // navbar-inverse kan bijv ook

</div>

Voor bootstrap objecten worden in het algemeen 2 classes gebruikt: één die aangeeft wat voor soort object het betreft (‘navbar’) en één die de styling aangeeft (‘navbar-default’).

Zet binnen de navbar <div> het volgende:

<div class=”container”>

<div class=”navbar-header”>

<a href=”” class=”navbar-brand”>My Website</a>

</div>

<div class=”collapse navbar-collapse”>

<ul class=”nav navbar-nav”>

<li><a href=””>Page 1</a></li>

<li><a href=””>Page 2</a></li>

<li><a href=””>Page 3</a></li>

<li><a href=””>Page 4</a></li>

</ul>

</div>

</div>

De ‘nav’ classes zijn allemaal standaard Bootstrap classes. De ‘navbar-header’ staat links in de navbar, de ‘collapse’ met de links in het midden. Wanneer je het browser window smaller maakt dan verdwijnen de links op een gegeven moment (breedte < 768 pixels), omdat de button die ervoor in de plaats komt nog niet is gemaakt.

Voeg binnen de navbar-header div het volgende toe:

<button type=”button” class=”navbar-toggle” data-toggle=”collapse”

data-target=”.navbar-collapse”> //”.” Niet vergeten!

// volgende regel is voor screenreaders

<span class=”sr-only”>Toggle navigation</span>

// de drie streepjes in het icon

<span class=”icon-bar”></span>

<span class=”icon-bar”></span>

<span class=”icon-bar”></span>

</button>

Het data-target is de link naar de list items, deze worden weergegeven als op de button wordt gedrukt.

Tot slot laat Rob zien hoe je kunt aangeven welke pagina uit de collapse list de actuele pagina is. Dit doe je door aan het list item de class “active” toe te voegen:

<li class=”active”><a href=””>Page 2</a></li>

**Les 147 – Forms and Tables**

Als je wilt dat content profiteert van de Bootstrap look and feel, dan moet je dat in een container <div> opnemen. Anders is het statische content die niet reageert op verbreden/verkleinen van de schermbreedte.

Creëer een container div en zet er een <h1> element en een invulformulier met één invulveld in:

<div class=”container”>

<h1>Our Bootstrap page</h1>

<form>

<input type=”text” />

<form>

<div>

Het invulformulier is nog heel erg basic Bootstrap. Rob stopt er wat meer Bootstrap in:

<form>

<div class=”form-group”>

<label for=”username”>Username</label>

<input type=tekst” class=”form-control” name’=”username” />

</div>

</form>

Het invoerveld wordt nu groter/kleiner als de schermbreedte wordt vergroot/verkleind. Rob adviseert een label te gebruiken voor de beschrijving van het invoerveld. Verder krijgt het invoerveld een naam “username”, waar vanuit het label aan wordt gerefereerd. De “name” is nodig bij het versturen van de ingevulde gegevens.

NB. Elk invoerveld moet in een aparte <div> met class “form-group” worden gestopt voor de optimale uitlijning.

Voeg nu een tweede invoerveld “password” toe onder het invoerveld:

<div class=”form-group”>

<label for=”password”>Password:</label>

<input type=”password” class=”form-control” name=”password” />

</div>

Vervolgens voegt Rob een check button toe:

<div class=”checkbox”>

<label>

<input type=”checkbox”>Keep me logged in

</label>

</div>

Het label omvat in dit geval het de checkbox. Dit moet om de styling netjes te houden.

Tot slot voegt Rob een Submit button toe. Dit kan met een <button> of een <input> veld. Rob kiest voor het laatste:

<div class=”form-group”>

<input type=”submit value=”Log in” class=”btn btn-default” />

</div>

Het formulier kan ook ‘inline’ gemaakt worden. Dit houdt in dat bij voldoende ruimte de velden naast elkaar op het scherm staan:

<form class=”form-inline”>

Wordt het scherm te smal, dan worden de invoervelden weer onder elkaar getoond.

Wil je het label links van het invoerveld, dam moet de form class “horizontal” gebruikt worden. Echter dit één op één zo invullen geeft niet het gewenste resultaat. Het lijkt alsof de invoervelden in een <div> met class=”sm-col-10” oid gewrapped moeten worden. Dit blijkt na controle inderdaad zo te zijn.

Vervolgens laat Rob diverse typen invoervelden zien. Bij type=”date” moet een datum ingevuld worden. Rechts in het invoerveld staat een datepicker ikoon. Die geeft Bootstrap er meteen bij, heel handig.

Bij type=”email” blijft het invoerveld er standaard uitzien. Druk je echter op de Log in (= submit) knop dan verschijnt een foutmelding als de ingevulde tekst niet het formaat van een emailadres heeft.

Andere input typen zijn op de site van getBootstrap.com te vinden.

De **input-groups** slaat hij echter over, hiermee kun je elementen, zoals een tekst of een icon, aan een invoerveld toevoegen. Dit moet dan in een input-group gewrapped worden zodat het direct naast elkaar op de pagina komt te staan:

<div class=”input-group”>

<span class=”input-group-addon” id=”myId”>@</span>

<input type=”text” class=”form-control” placeholder=”Username”

area-described-by=”myId”>

</div>

Het tweede onderwerp van deze les is tabellen. Maak een standaard HTML tabel aan binnen de container <div> (let hier op!):

<table class=”table”> // de std Bootstrap table class tbv styling

<tr>

<th>Voornaam</td> // Header row

<th>Achternaam</td>

<th>Leeftijd</td>

</tr>

<tr>

<td>Frank</td> // Cell data

<td>de Groot</td>

<td>51</td>

</tr>

<tr>

<td>Anita</td>

<td>Peters</td>

<td>47</td>

</tr>

</table>

De table class “table” zorgt dat de styling netjes door Bootstrap wordt verzorgd. ER zijn diverse klassen die op tables, rows en cells kunnen worden toegepast, bijv:

* table class: “table-striped”, “table-bordered”, table-hover”, etc
* table row en cell: “success”, “danger” verandert de achtergrondkleur van de cel in lichtgroen, resp. lichtrood.

**Les 149 – Bootstrap Components**

In deze les worden een aantal componenten in Bootstrap, die het leven van de web developer veraangenamen, besproken. Zie hiervoor de Components sectie op getBootstrap.com.

NB Let op dat de icons in de ../fonts directory staan, anders gaat dit niet werken.

Rob start met de glyphicons, dat zijn allerlei voorgedefinieerde icons die je eenvoudig kunt gebruiken door een <span> in je code op te nemen:

<span class=”glyphicon glyphicon-search”></span>

Omdat de glyphicons een font elementen zijn kunnen ze naar smaak gestyled worden, bijv:

.large {

font-size:800%;

color:red;

}

<span class=”glyphicon glyphicon-thumbs-up large”></span>

Er ontstaan geen rafelige randjes oid, omdat het een font betreft.

De glyphicons kunnen worden gebruikt als onderdeel van inputs, zoals buttons en formulieren:

<div class=”input-group”>

<span class=”input-group-addon glyphicon glyphicon-eur”></span>

<input type=”text” class=”form-control” />

</div>

Een glyphicon kan ook in een Bootstrap button worden opgenomen, door de <span> binnen de <button> te plaatsen:

<button type=”button” class=”btn btn-success”>

<span class=”glyphicon glyphicon-eur”>Amount in</span>

</button>

Alerts kent Bootstrap ook:

<div class=”alert alert-success”>Het is je gelukt!</div>

Dit creëert niet zozeer een alert alswel een tekstbox met lichtgroene achtergrond, die niet weggeklikt kan worden. Wil je de melding wel weg kunnen klikken dan moet het volgende er nog bij:

<div class=”alert alert-success **alert-dismissable**”>Het is je gelukt!

<button type=”button” class=”close” data-dismiss=”alert”>

x<button>

</div>

Er verschijnt nu een ‘x’ rechts in de alert. Klik je daarop, dan verdwijnt de alert van het scherm.

Dan laat Rob een voorbeeld van een list-group en badges zien:

<ul class=”list-group”>

<li class=”list-group-item”>Item 1</li>

<li class=”list-group-item”>Item 2</li>

<li class=”list-group-item”>Item 3</li>

<li class=”list-group-item”>Item 4</li>

</ul>

Voeg een badge toe aan een list-item:

<li class=”list-group-item”>Inbox

<span class=”badge”>23</span>

</li>

**Les 151 – JavaScript Components**

In deze les behandelt Rob een aantal javascript componenten, voor meer interactie met de website.

De eerste is de ‘modal’ component, dit is een pop-up window dat verschijnt en de rest van de pagina blokkeert. Wordt gebruikt voor log-in formulieren, om info te tonen etc. Voeg de volgende code toe binnen een container <div>:

<div class=”container”>

<div class=”modal” id=”myModal”>

TEKST

</div>

</div>

Er verschijnt niets omdat een modal standaard verborgen is. Daarvoor is een button nodig, voeg deze vóór de modal div toe (erna werkt niet!):

<button class=”btn btn-success” data-toggle=”modal” data-target=”#myModal”>

Launch Modal

</button>

Door op de button te drukken verschijnt er nu een leeg modal scherm (tekst wordt niet getoond), dat weer weggeklikt kan worden door op een willekeurige plek op de pagina te klikken.

Content kan worden toegevoegd en zichtbaar gemaakt door een aantal door Bootstrap vereiste <div>s toe te voegen:

<div class=”modal” id=”myModal”>

<div class=”modal-dialog”>

<div class=”modal-content>

<div class=”modal-header”>

<button class=”close” data-dismiss=”modal”>x

</button>

<h4 class=”modal-title”>Titel van mijn modal</h4>

</div>

</div>

</div>

<h4> wordt meestal gebruikt voor modal titels. De close button wordt in de modal getoon (hier als ‘x’). De modal verschijnt nu met de header titel. Hij kan worden weggeklikt door op het kruisje of buiten de modal te klikken.

Voeg nu de modal body met wat tekst toe, op hetzelfde <div> niveau als de modal header:

<div class=”modal-body”>

Dit is de inhoud van mijn modal

</div>

Tot slot voegt Rob de modal footer toe, deze wordt meestal gebruikt om wat buttons toe te voegen:

<div class=”modal-footer”>

<button class=”btn btn-default” data-dismiss=”modal”>Close</button>

<button class=”btn btn-success”>Save changes</button>

</div>

De modal kan nu ook met de Close button gesloten worden.

De grootte van de modal kan nog ingesteld worden door een class toe te voegen aan de modal-dialog <div>:

<div class=”modal-dialog modal-sm”>

**Les 153 – ScrollSpy**

Met ScrollSpy biedt Bootstrap de mogelijkheid het scherm te scrollen, waarbij een icon bovenaan het scherm aangeeft hoeveel je ongeveer gescrolled hebt.

Voeg daartoe de volgende gekleurde vierkanten toe:

.contentDiv {

height:800px;

}

#div1 {

background-color:red;

}

#div2 {

background-color:green;

}

#div3 {

background-color:blue;

}

<div id=”div1” class=”contentDiv”>

<div id=”div2” class=”contentDiv”>

<div id=”div3” class=”contentDiv”>

De navigatiebar verdwijnt standaard van het scherm wanneer gescrold wordt. Dit voorkom je door de navbar een class”navbar-fixed-top” te geven (oftewel position fixed):

<div class=”navbar navbar-default navbar-fixed-top”>

Doel hiervan is om in de navigatiebar de scroll positie weer te geven, maar dan moet de navbar natuurlijk wel zichtbaar zijn.

De list items uit de navbar worden gebruikt om naar div1, div2 en div3 te navigeren:

<div class=”collapse navbar-collapse”>

<ul class=”nav navbar-nav”>

<li><a href=”#div1”>Page 1</a></li>

<li><a href=”#div2”>Page 2</a></li>

<li><a href=”#div3”>Page 3</a></li>

</ul>

</div>

Om ScrollSpy toe te voegen moet aan de <body> tag het volgende toegevoegd worden:

<body data-spy=”scroll” data-target=”.navbar-collapse”>

De data-target moet wijzen naar de plek waar de interne links zijn opgenomen.

**Les 153 – Project – Landing page introduction**

In deze les vat Rob samen wat in de voorgaande Bootstrap lessen behandeld is en geeft hij de aanzet voor het project App Landing Page: bouw je eigen landing page voor een app of wat dan ook mbv Bootstrap.

**Les 154 – Project – App Landing Page**

Rob begint met de navigation bar. Hij maakt vrijwel dezelfde bar als in les 145 is gebouwd, met een paar afwijkingen/uitbreidingen.

De navbar header is gelijk aan wat eerder gemaakt is.

De navbar collapse is iets uitgebreider omdat Rob er een invulformulier voor aanloggen bij stopt, deze wordt binnen de container <div> in de navbar <div> opgenomen, direct na de navbar-header <div>:

<div class=”collapse navbar-collapse”>

<ul class=”nav navbar-nav”>

<li class=”active”><a href=”#home>Home</a></li>

<li><a href=”#about”>About</li>

<li><a href=”#download”>Download</li>

</ul>

<form class=”navbar-form navbar-right”>

<div class=”form-group”>

<input type=”email” placeholder=”Email”

class=”form-control” />

</div>

<div class=”form-group”>

<input type=”password” placeholder=”Password”

class=”form-control” />

</div>

<button type=”submit” class=”btn btn-success”>Log In

</button>

</form>

</div>

Door de placeholder te gebruiken wordt ruimte op de pagina bespaard omdat een label nu niet meer nodig is.

Van de navbar-brand <div> met de naam van de app maakt Rob het font wat groter door de styling aan te brengen:

.navbar-brand {

font-size:1.8em;

}

Voor de navbar-brand kan ipv platte tekst ook een plaatje opgenomen worden. Vervang de tekst dan door:

<img alt=”Brand” src=[dir link naar plaatje] />

NB alt=naam voor plaatje die getoond wordt als het plaatje niet kan worden getoond.

Voor het middenstuk van de pagina gebruikt Rob een foto als achtergrond, daar komt nog het nodige bovenop. De manier om dat te doen is een container <div> te creëren en deze te stylen met een achtergrondplaatje:

<style>

#imageContainer {

background-image:url(“images/Provence.jpg”);

height:400px;

}

</style>

<div class=”container” id=”imageContainer”>

</div>

Let op dat de hoogte initieel is ingevuld anders zie je niets. Er verschijnt nu echter maar een klein deel van de foto. Allereerst zetten we de hoogte van de foto op 100%, dwz gelijk aan de schermhoogte. Dit werkt niet door simpelweg de styling op 100% te zetten (geen idee waarom eigenlijk). Rob doet dit dmv een stukje jQuery, zoals in dat hoofdstuk ook gedaan is:

$(“#imageContainer”).css(“height”, $(window).height());

Er wordt nu meer van de foto getoond, echter staat de foto er niet helemaal op en zit er nog de nodige padding. De padding halen we weg door de imageContainer een breedte van 100% te geven in de styling:

width:100%;

Nu wordt de foto in mijn geval horizontaal en vertikaal herhaald om de ruimte op te vullen. Dit lossen we op door

image-repeat:no-repeat;

in de imageContainer style op te nemen. Nu ontstaan er echter weer witte delen omdat de foto te klein is voor het volledige scherm. CSS heeft hiervoor een truc:

background-size:cover;

Voeg je dit toe dan wordt het scherm volledig gevuld met de foto, in ieder geval in de breedte. Bij Rob gaat het scherm knipperen als hij het browserwindow versmalt en verbreedt, ik heb daar geen last van.

In de container <div> met de achtergrond foto gaat Rob content toevoegen. Hij start met en 6 kolommen brede <div> die gecentreerd wordt getoond:

<div class=”row”>

<div class=”col-md-6 col-md-offset-3”>

<h1>Fietsvideos</h1>

</div>

</div>

De tekst wordt nu echter boven de achtergrond foto getoond onder de navbar en is niet zichtbaar. De tekst moet naar beneden. Hiervoor voegt Rob een id toe aan de <div>:

< div class=”col-md-6 col-md-offset-3” id=topRow”>

#topRow {

margin-top:100px; // 100px naar beneden

tekst-align:center;

font-size:300%;

}

De tekst wordt nu wat lager weergegeven, maar de font-size heeft geen effect. Deze stellen we in de <h1> style in:

#topRow h1 {

font-size:400%;

}

Vervolgens voegt Rob tekst toe die zijn app omschrijft. Hiervoor gebruikt hij de Bootstrap class ‘lead’, die een <p> net wat groter maakt:

<p class=”lead”>Korte omschrijving van het product”</p>

<p class=”lead”>Meer uitgebreide omschrijving”</p>

Dan nog een <p> om je aan te melden voor de mailing list, deze maakt Rob bold:

<p class=”’bold”>Geinteresseerd? Meld je aan voor onze mailing list!</p>

.bold {

font-weight:bold;

}

Hieronder komt een invoerveld waarin je het emailadres voor de mailing list kan invoeren met eronder een submit button. Hiervoor creëert Rob een <form>:

<form>

<div class=”form-group> // 🡺 Kan weg, niet nodig!

<div class=”input-group”>

<div class=”input-group-addon”>@</div>

<input type=”email” class=”form-control”

placeholder=”Jouw e-mail adres” />

</div>

</div>

<input type=”submit” class=”btn btn-success btn-lg” />

</form>

Let op dat deze form nog binnen de ‘col-md-6 <div> zit.

NB De form-group <div> hebben we niet nodig omdat we geen labels bij de inputvelden gebruiken, kan dus weg uit de code.

NB de class’btn-lg’ maakt de button groter.

Vervolgens creëert Rob een tweede container met meer tekstuele info over de app. Met een paar extra styling classes ziet het er net wat beter uit:

.center {

tekst-align:center;

}

.title {

margin-top:100px;

font-size:bold;

}

<div class=”container contentContainer”>

<div class=”row center”>

<h1 class=”title”>Meer informatie</h1>

<p class=”’lead center”>Meer informatie over de videos volgt nog

</p>

</div>

</div>

De contentContainer class gebruiken we in het stukje javaScript waarmee voorheen de hoogte van de id imageContainer werd aangepast:

$(“.contentContainer”).css(“min-height”, $(window).height());

Rob kiest voor min-height omdat de foto er dan ook goed uitziet wanneer de hoogte van het window kleiner is dan de hoogte van de foto.

Binnen deze container maakt Rob een nieuwe row met 3 even brede kolommen. Hierin komt nog meer tekst, per kolom bestaande uit een header, paragraaf en een button:

<div class=”row”>

<div class=”col-md-4 marginTop”>

<h1><span class=”glyphicon glyphiconX”></span> Kolom1

</h1>

<p>Vul hier de nodige tekst in</p>

<button class=”btn btn-success marginTop”>

Meld je aan!</button>

</div>

<div class=”col-md-4 marginTop”>

<h1><span class=”glyphicon glyphiconY”></span> Kolom2

</h1>

<p>Vul hier de nodige tekst in</p>

<button class=”btn btn-success marginTop”>

Meld je aan!</button>

</div>

<div class=”col-md-4 marginTop”>

<h1><span class=”glyphicon glyphiconZ”></span> Kolom3

</h1>

<p>Vul hier de nodige tekst in</p>

<button class=”btn btn-success marginTop”>

Meld je aan!</button>

</div>

</div>

Bij de header zet Rob nog een glyphicon, vul voor de X een beschikbare code in.

Dan volgt de derde container die als footer wordt ingevuld. Hierin wordt aangespoord de app te downloaden. De container krijgt een lichtblauwe achtergrondkleur, een kopregel en een button met een plaatje (Rob maakt er een <input> van):

#footerContainer {

background-color:lightblue;

width:100%;

}

# downloadFromAppStoreButton {

border-radius:5px;

margin-top:20px;

}

<div class=”container contentContainer” id=”footerContainer”>

<div class=”row center”>

<h1 class=”title”>Download de videos vandaag nog!</h1>

<button type=”button” id=”downloadFromAppStoreButton”>

<img src=”images/download\_from\_app\_store\_icon.png”>

</button>

</div>

</div>

Omdat de buttons vaak rechthoekige plaatjes zijn moeten de hoeken afgerond worden om de witte hoeken weg te krijgen.

**Section 6 - WordPress**

**Les 159 – What is WordPress?**

WordPress is een CMS, waarmee een website ontworpen en onderhouden kan worden. Oorspronkelijk bedoeld om blogs te maken. Is gebouwd op PHP en MySQL, waarmee het naar wens aangepast en uitgebreid kan worden. Een WordPress website kan zonder codeer kennis gebouwd worden.

Op de **Wordpress.org** website is de nodige info te vinden. Daar kan WordPress ook gedownload worden. Echter bieden veel hosting providers – zoals EcoWebhosting – een Wordpress installatiescript. Ga hiervoor naar je domains op EcoWebhosting en klik het domain aan waarmee je wil gaan werken. Het Control Panel ervan verschijnt nu. Ga nu een stuk naar beneden en klik op **Wordpress Blog** in CGI Scripts. De WordPress pagina verschijnt nu. We gaan een ‘single blog’ installeren in de dir ‘wordpress/’, vul die naam in in het invoerveld bij Installation Directory. Je kunt het ook in de root van je domein installeren, maar dat doen we hier dus niet.

Na installatie verschijnt een scherm met domeinnamen. Rob gebruikt de tijdelijke domeinnaam, de andere zou je moeten kopen om ze te kunnen gebruiken (en dat heb ik nog steeds niet gedaan). In het scherm dat nu verschijnt heb ik als Site Title ‘Wordpress Demo’, als username ‘famasoft’ en als pws ‘E..98’ ingevuld.

Rob waarschuwt dat Wordpress sites veelvuldig worden gehackt. Het password dat je kiest moet daarom zo ingewikkeld mogelijk zijn (het mijne is dat zeker niet) en zodra een nieuwe Wordpress release uitkomt moet je je site daarmee bijwerken anders is de kans groot dat je site gehackt wordt.

Na installatie verschijnt een Wordpress inlogscherm. Na invullen van usernaam en password verschijnt het Wordpress control panel. Hier kun je je site ontwerpen, aanpassen en alle administratie eromheen doen. Als je de site die je aan het bouwen bent wilt bekijken, klik dan linksboven op Wordpress Demo en ‘visit site’.

**Les 160 – The Wordpress Dashboard**

Vanuit het Dashboard doe je alles met je website. Rob begint met een uitleg over Posts, omdat Wordpress oorspronkelijk is opgezet als blog platform. Er staat nu één post in. Deze kun je aanpassen, zowel de header als de inhoud, welke visueel of als html-tekst kan worden aangepast. Ook de permalink kan aangepast worden, Rob kiest voor het wat vriendelijker ‘**post name’** formaat.

Wanneer je veel post is het handig om de posts in **Categories** onder te verdelen, waar op gezocht kan worden. Hiermee breng je structuur aan in je posts.

Met **tags** kun je het zoeken naar posts ook vereenvoudigen. Het verschil met categories is dat hier geen structuur in zit. Een post maakt in principe onderdeel uit van één catagory maar kan diverse tags hebben.

Media kan geupload worden op je site en vervolgens in posts gebruikt worden. Wordpress is goed in het resizen van images, handig als je niet de volledige grootte nodig hebt.

**Pages** zijn bedoeld om statische content weer te geven, zoals een About page etc etc. Deze staan los van de posts.

**Comments** kun je toestaan per post of pagina.

**Appearance** laat je het uiterlijk van de website aanpassen, meer daarover in de volgende les.

**Plugins** zijn een krachtig tool van Wordpress. Komt later terug.

**Users** staat je toe user toe te voegen en rollen er aan toewijzen.

**Tools** zullen we enkel gebruiken om content te importeren.

Vwb de **settings** noemt Rob een aantal punten:

* de email setting staat je toe om automatisch berichten te posten als je een mail naar het hier ingevulde adres stuurt. Hou dit adres daarom geheim!
* Onder Reading stelt Rob standaard altijd ‘static page’ in bij ‘front page displays’.

Vervolgens laadt Rob een demo Wordpress site. Deze haalt hij van **wpcandy.com/made** (let op: ik kan hier op de wpcandy site zelf niet bijkomen, dus gebruik de link). Download de zip file bij Easier Theme Development With Sample Wordpress Content. Ga vervolgens naar Tools – Import in het Dashboard en selecteer Wordpress. Installeer en activeer daarna de Wordpress Importer (als nog niet gedaan is) en laadt de posts.xml file die van wpcandy.com gedownload is. Kies vervolgens de standaard instellingen om ermee te werken.

Rob loopt nu door de posts heen en wijst op het menu boven aan het scherm, dat aanpasbaar is. Dit kan in het Dashboard vanuit Appearance – Menus. Hij maakt een nieuw menu aan en zet hier de pagina’s in die hiervandaan bereikt kunnen worden. De structuur van het menu kan naar wens aangepast worden. In Manage Locations geef je aan wat het primary en secondary menu is. Stel de primary in op het zojuist gecreëerde menu en doe een Visit site.

Wordpress heeft standaard een site-zoekmachine ingebouwd. Dit is heel handig omdat het lastig zelf te maken is.

**Les 161 – Installing Themes**

Selecteer in het Dashboard de optie Weergave en dan de optie Thema’s. Er verschijnen nu een aantal thema’s waarvan er één uitgekozen kan worden. Rob kiest voor het thema Sugar & Spice. Met Visit Site kan het effect bekeken worden. Het thema kan ook naar wens aangepast worden met optie Thema Opties uit het Weergave menu.

Vanuit Wordpress zijn de nodige gratis thema’s beschikbaar, deze kunnen tot op zekere hoogte naar wens ingesteld worden maar heel ver gaat dat niet. Wil je echt iets bijzonders van je website maken dan zul je een premium thema moeten kopen, deze zijn in het algemeen veel verder aan te passen. Rob gaat hiervoor naar de site **themeforest.net,**  de grootste provider van themes. Rob maakt veel gebruik van de twee best verkochte thema’s: **Avada** en **X**, waarvan X enorm veel aanpassingsmogelijkheden heeft. Deze kost 63$ maar via Rob kun je 20% (10$) korting krijgen. Met X thema kunnen compleet anders uitziende websites worden gebouwd, zit vol met animiaties en er zit zelfs een webstore WooCommerce bij.

**Les 162 – Premium Themes**

Rob gaat met het premium X thema de website mooier maken. Deze schaft hij aan bij **themeforest.net.** De download resulteert in een lokale directory ‘x\_package’ (naam bij Rob) waarin de x.zip file staat. Deze bevat de content die naar Wordpress geupload moet worden. Ga hiervoor naar de Wordpress dashboard en selecteer Appearance-Themes (Weergave – Thema’s). Selecteer vervolgens ‘Add new’ (Nieuwe toevoegen) en daarna ‘Upload Theme’ (Thema uploaden). Zoek de x.zip file en selecteer Install Now. Wanneer het thema geinstalleerd is activeer het dan direct.

Premium thema’s hebben meestal een aantal plug-ins nodig om goed te kunnen werken. Dit wordt na installatie aangegeven. Rob installeert simpelweg alle plug-ins. De X-shortcode plug-in mislukt, deze download hij van de X-theme site, waar je alleen bij kunt als je het X-theme hebt gekocht. Er is in de tussentijd het nodige veranderd, want ik zie dit niet dus ga ervan uit dat dit niet nodig is (?).

Rob laat zien hoe je de instellingen van een theme kunt aanpassen. Dit doe je door in het Weergave de optie Customizer te selecteren. Je kunt hier allereerst één van vier stacks kiezen, waarmee je de basis lay-out van je website instelt. Daar kun je vervolgens het nodige van instellen, zoals max. schermbreedte, achtergrondkleur etc etc.

Vervolgens gaat Rob demo content laden via de Add-ons optie bovenin het dashboard. Hij kiest voor de Renew5 add-on. Hiermee pas je feitelijk de instellingen van het gekozen thema in één klap aan, dus je hoeft niet alle instellingen met de hand te wijzigen. In Rob’s geval moet hij files downloaden, maar het lijkt er op dat dit inmiddels heel anders gaat met het X theme. In mijn geval kon ik de settings selecteren en hoefde niet meer te doen. Het eindresultaat was hetzelfde.

Tot slot past Rob het menu aan via dasboard optie Weergave – Menu’s. Hier kun je zelf menu’s samenstellen en deze als primary en/of footer menu gebruiken.

**Les 163 – Creating a blog**

Allereerst past Rob de titel van de blog aan via de X-theme Customizer.

Dan verwijdert Rob de eerste Post via dashboard optie Berichten – Alle berichten. Hier kun je berichten verwijderen, aanpassen en toevoegen. Wijst zich vanzelf.

Vervolgens gaat Rob widgets toevoegen aan de hoofdpagina, via dashboard menu Weergave – Widgets. Dit is een kwestie van de widgets naar de beschikbare ruimten slepen. Rob zet bijv. een search box in de main side bar.

Daarna gaat Rob een contact formulier maken. Hiervoor gebruikt hij de plug-in Contact Form 7 die bij hem al aanwezig is. Bij mij niet, ik heb deze als nieuwe plug-in opgezocht en geïnstalleerd, via het dashboard menu Plugins. De contact form plug-in bevat een zg. shortcode die rechtstreeks op in een pagina kan worden geladen, als ware het HTML code. Zoek de shortcode op, knip deze en plak vervolgens op de Contact Us pagina, vanuit het dashboard via Pagina’s.

De Contact Us pagina bevat default een ‘leave a reply’ deel. Als je daar van af wil open dan opnieuw de Contact Us pagina en selecteer Bewerken. Open de scherrminstellingen boven in het scherm en selecteer Reacties (de linker in mijn geval). Onder aan het scherm verschijnen nu de bijbehorende Reacties opties. Deselecteer hier ‘Reacties toestaan’ en toon de Contact Us pagina opnieuw. Het ‘leave a reply’ deel is nu verdwenen.

Wil je reacties in het algemeen uitzetten, dus niet voor één specifieke pagina, dan kan dat via de algemene Instellingen en selecteer hieruit Reacties. Vink de optie ‘Sta toe dat bezoekers kunnen reageren op nieuwe artikelen’ af en klaar is Kees.

**Les 164 – Creating an eCommerce site**

In deze les bouwt Rob een webshop. Hij gebruikt hiervoor de demosite met de blog uit de vorige les en voegt daar een webshop aan toe. Hiervoor gebruikt hij de gratis **WooCommerce** plugin. Deze integreert ook goed met het X theme.

Rob download de WooCommerce plugin en laat de nodige instellingen zien, zonder deze te veranderen. Plugins zijn in essentie een verzameling php files. Wil je de plugin veranderen, dan doe je dat daar (zeer riskant!). Rob laat dan de **dummy-data.xml** file zien. De inhoud ervan kopieert hij in een nieuwe file in TextWrangler (via cut & paste). Hij bewaart de file en noemt hem dummy-data.xml. Deze importeert hij vervolgens in WordPress via dashboard Extra – Importeren, en kiest dan voor WordPress. Vóór het downloaden zet hij de optie ‘Download and import file attachments’ aan, wat tot gevolg heeft dat foto’s etc ook geïmporteerd worden. Hiermee importeer je een dummy webshop met de nodige dummy producten.

Rob laat zien hoe je een shortcode hierbij kan gebruioken. Op één van de shop pagina’s laadt Rob de ’12 most recent items’ shortcode, hij haalt deze van **docs.woothemes.com**, maar er zijn genoeg sites hiervoor te vinden. Hij laadt de short code voor de 12 meest recente items en zet deze op de Shop pagina.

Tot slot laat Rob zien hoe je een ander thema downloadt en installeert, waardoor de shop er ineens heel anders uit ziet. Dit gaat bij hem nog redelijk ingewikkeld, er moet bijv. een zip file gedownload worden die weer geïmporteerd wordt. Ik kan via de X Add-ons uit het dashbord direct een thema selecteren en toepassen.

**Les 166 – Further reading**

In deze les worden de nodige links gegeven om verder te stoeien met Wordpress. De officiële site is wordpress.org.

**Section 7 – PHP**

**Les 168 – What is PHP?**

PHP is bedoeld voor het bouwen van server-side programma’s, de back-end, waar de business logic zich bevindt, voor bijv het ophalen van gevraagde gegevens.

On **php.net** is de nodige info te vinden, waaronder de geschiedenis. De oorspronkelijke betekenis van PHP is Personal Home Page tools, maar dat is door de jaren heen veranderd in de recursieve definitie PHP: Hypertext Processor.

Om PHP programma’s te draaien heb je een webserver nodig: gebruik die op EcoWebhosting.

**Les 169 – Your first PHP script – Hello World**

In deze les wordt het eerste Hello World php programma gemaakt. Dit doet Rob op de webserver van Ecowebhosting, maar het kan ook met een lokale IDE zoals ZendServer, zie in PHP boek voor de nodige alternatieven.

Mbv FileZIlla een dir /php aanmaken in de /public folder. Maak met TextWrangler

**test.php** file aan en zet hier de tekst Hello World in. Zet de file met FileZilla over naar de php folder op EcoWebhosting. Run de file nu door

famasoft.nl/public/test.php

in de browser in te typen. NB Zorg dat famasoft.nl naar de juiste folder bij EcoWebhosting wijst, **176.32.230.43/famasoft.nl/** in dit geval.

Maak nu een echte php file die als zodanig door de browser wordt herkend:

<?php

echo “This is my first PHP program”;

?>

**Les 171 – Displaying error mesages**

Javascript code kun je debuggen mbv de Console in de browser. Omdat php code op de server draait kan dat niet op die manier. Dat werkt anders, maak hiervoor een **php.ini** file aan op de webserver in de /php directory. Dit is een config file die door de server wordt gecheckt op specifieke settings wanneer php code wordt uitgevoerd. De errorlogging staat standaard uit, dat wil je nl. niet op productie websites. Zet de volgende settings in php.ini:

display\_errors = on

display\_startup\_errors = on

Maak nu een syntax fout aan in de test.php file, laat bijv de afsluitende ” weg. Run test.php en er verschijnt een Parse error in de browser.

**Les 172 – PHP variables**

Variabelenamen beginnen in php altijd met een $ teken:

$test = “I am a variable”;

echo $test;

Een “ teken krijg je in de string door er een \ voor te zetten:

$test = “Dit is \”mooi\””;

Strings worden dmv een . (dot) aan elkaar geplakt:

$test1 = “I am “;

$test2 = “ a variable”;

echo $test1 . $test2;

Een numerieke variabele:

$number = 75;

echo $number / 3;

Strings en numerieke variabelen kunnen zonder meer aan elkaar geplakt worden:

echo $test1 . $test2 . $number;

En een variabele kan zo maar van type veranderen:

$test1 = “String variabele”;

$test1 = 75;

Een variabele kun je ook zonder meer in een string opnemen:

$sname = “Frank”;

echo “My name is $name”;

Dus hoeft dus niet perse met de . concatenator.

Een leuk en soms handig trucje is om ‘variabele variabelen’ te gebruiken, waarbij de waarde van een (string)variabele wordt gebruikt om een andere variabele te identificeren:

$name = “Frank”;

$a = “name”;

echo $$name;

Door zelf de variabele te substitueren zie je al dat hier “Frank” getoond zal worden.

NB Line breaks worden dmv de <br /> html tag ingevoegd:

echo “<br /><br />”;

**Les 174 – Arrays**

Arrays kunnen op verschillende manieren in php gedefinieerd worden:

$myArray = array(“pizza”, “chocola”, “”koffie”);

Is de eenvoudigste manier. De waarden in een array kunnen niet mbv “echo myArray” worden getoond. Hiervoor kan **print\_r** gebruikt worden:

print\_r($myArray);

Dit zal alle voorkomende elementen van de array tonen.

Een element wordt als volgt benaderd:

echo myArray[2];

zal het derde element tonen (index start met 0). Het opvragen van een niet bestaand element levert geen foutmelding op in php, bijv:

echo myArray[5];

toont een lege string. In dit opzicht is php makkelijker in de omgang dan javascript.

Een andere manier om een array te definiëren is om een index van een waarde te voorzien:

$anotherArray[0] = “pizza”;

$anotherArray[1] = “chocola”;

De variabele hoeft niet eerst gedeclareerd te worden.

Arrays hoeven niet perse numerieke index te hebben, zoals hieronder is te zien:

$thirdArray = array(

“France” => “French”,

“USA” => “English”,

“Netherlands” => “Dutch”

);

De landnamen zijn de indexes in dit geval.

Een array element wordt als volgt als laatste element aan een bestaande array toegevoegd:

$anotherArray[0] = “pizza”;

$anotherArray[1] = “chocola”;

$anotherArray[] = “salad”;

Dit zal een element met index 2 toevoegen. Handig als je niet weet hoe groot een array is.

Een array element wordt dmv de **unset** methode verwijderd:

unset($thirdArray[“USA”]);

De elementen na [“USA”] schuiven alle een plaatsje terug.

De unset methode werkt ook op gewone variabelen, bijv.

$name = “Frank”;

echo $name;

unset $name;

echo $name;

De tweede echo zal niets meer tonen.

**Les 175 – If statements**

De syntax is identiek aan die in javascript:

$number = 1;

if ($number == 1) {

echo “equal to 1”;

} else {

echo “not equal to 1”;

}

Ongelijkheid test:

$number = 1;

if ($number != 1) {

echo “not equal to 1”;

} else {

echo “equal to 1”;

}

Combinatie van vergelijkingen is anders dan in javascript:

if ($a==$b AND $a==$c) {

// statements

}

Kan ook met OR.

**Les 176 – For and foreach**

Syntax lijkt erg op javascript:

for ($i=1; $i,=10; $i++) {

// statements

}

NB De variabele hoeft niet zoals in javascript met een var gedeclareerd te worden.

Met de for each kan door een array gewandeld worden:

$myArray = array(“cat”, “dog”, “mouse”, “bird”);

foreach ($myArray as $key => $value) {

echo “Key: $key, value: $value <br />”;

}

**Les 178 – While loops**

Dit lijkt als twee druppels op javascript:

$i = 1;

while ($i < 20) {

echo $i;

$i++;

}

Door een array lopen totdat het laatste element geweest is:

$i=0;

$myArray = array(“apple”, “banana”, “grape”);

while ($array[$i]) {

echo $array[$i] . “<br />”;

$i++;

}

**Les 179 – Sending email with PHP**

In de basis is voor het versturen van een email een ontvangstadres, een onderwerp, een inhoud en een afzender – als onderdeel van de header – nodig:

<?php

$emailTo = [casagrande2407@gmail.com](mailto:casagrande2407@gmail.com);

$subject = “Test email van Frank vanuit PHP”;

$body = “Inhoud van de test mail”;

$headers = “From: [fdegroot000@ziggo.nl](mailto:fdegroot000@ziggo.nl)”;

mail($emailTo, $subject, $body, $headers);

?>

En verrek, het werkt gewoon…. Geen gedoe met instellen van mail servers, hoe kan dat?

De **mail** methode geeft een waarde (1 bij succes) terug die gebruikt kan worden om te bepalen of de mail daadwerkelijk is verstuurd.

if (mail($emailTo, $subject, $body, $headers)) {

// succes code

} else {

// fout code

}

NB Het succesvol versturen van de mail betekent niet dat deze ook succesvol is ontvangen. Wanneer bijv. het mailadres niet klopt dan kan de mail toch gewoon met succes verstuurd worden. Maar is het emailadres veld niet met een mailadres ingevuld (bijv. leeg) dan krijg je wel een ‘0’ terug.

**Les 181 – GET variables**

Get variabelen worden via de url aan de browser doorgegeven, zoals:

[www.famasoft.nl/php/index.php?name=Frank](http://www.famasoft.nl/php/index.php?name=Frank)&surname=deGroot

Dit geeft de variabelen ‘name’ en ‘surname’ met de waarden ‘Frank’ resp. ‘deGroot’ door aan de browser. In PHP worden de variabelen in de system array **$\_GET** doorgegeven. De inhoud ervan kan met

print\_r($\_GET);

opgevraagd worden.

Het nadeel van de GET veriabelen is dat deze via de url doorgegeven worden, zichtbaar voor de gebruiker. Voor ongevoelige data geen probleem, maar voor usernames en passwords ongeschikt. Daarvoor worden de meer gebruikelijke POST variabelen gebruikt (zie volgende les).

Het zelf creëren van variabelen via de url is ongebruikelijk. Normaliter gebeurt dat via invoervelden. Daartoe bouw Rob een formulier met één invoerveld waar een usernaam ingevoerd kan worden. In de index.php file wordt nu html en php code door elkaar opgenomen:

<body>

<div>

<form>

<label for=”name”>Name</label>

<input name=”name” type=”text” />

<input type=”submit” />

</form>

</div>

</body>

Dit creëert een tekstinvoerveld met de naam “name” en een submit button. Wanneer op de submit button wordt gedrukt wordt de inhoud van het invoerveld “name” aan de browser doorgegeven en kan het via PHP code afgevangen worden. Voeg de volgende PHP code voor of na de html code toe:

<?php

if ($\_GET[“name”]) { // als variabele niet leeg is

echo “your name is “ . $\_GET[“name”];

}

?>

Deze code is nog niet af. Wanneer niets wordt ingevuld en op de submit knop wordt gedrukt dan gebeurt er niets, terwijl dat misschien wel de bedoeling is. Dit kunnen we oplossen door een variabele aan de submit knop te koppelen:

<input type=”submit” name=”mySubmit” value=”invoeren” />

In de submit knop wordt nu de tekst “invoeren” getoond. Wordt de knop ingedrukt dan komt de “mySubmit” variabele met de waarde “invoeren” mee en wordt in de $\_GET array gezet, ook wanneer de “name” variabele leeg is. Er kan nu worden getest op een leeg invoerveld:

if ($\_GET[“submit”]) { // als variabele “submit” niet leeg is

if ($\_GET[“name”]) { // als variabele “name” niet leeg is

echo “your name is “ . $\_GET[“name”];

} else {

echo “Geen naam ingevuld. Probeer opnieuw.”;

}

}

**Les 183 – POST variables**

Wil je de variabelen niet via de url doorsturen dan moet je POST variabelen gebruiken. Hiervoor moet de methode van de form op “post” gezet worden:

<form method=”post”>

// html code

</form>

De default form methode is “get”, vandaar dat we “post” expliciet moeten opgeven. Vervang de $\_GET door $\_POST in de code, verwijder de variabelen in de url en laadt de php file opnieuw. De werking is hetzelfde, alleen worden de variabelen nu niet meer in de url balk getoond.

De code wordt nu uitgebreid met de herkenning van een eerder aangemelde gebruiker. Hiertoe zetten we een aantal namen in een array namesArray, als de ingevoerde naam overeenkomt met één van de namen uit de array dan wordt een welkomstboodschap getoond:

$namesArray = array(“Frank”, “Anita”, “Marcella”, “Sophia”);

$bekendeGebruiker = 0;

// binnen de $\_POST[“name”]) :

foreach ($names as $name) {

if ($name == $\_POST[“name”]) {

echo “Welkom terug “ . $name;

bekendeGebruiker = 1;

}

}

if (!bekendeGebruiker) {

echo “Onbekende gebruiker”;

}

**Les 185 – Mini Challenge: Contact Form**

In deze les wordt een mooi ogend contactformulier gebouwd, bestaande uit naam, email en subject. Wanneer deze is ingevuld wordt er een email gestuurd naar de ingevulde afzender.

Start met de code van les 179 en zet deze in commentaar, anders wordt elke keer dat de pagina wordt ververst een mail verstuurd.

Rob gebruikt Bootstrap om het formulier te creëren. Neem de Bootstrap links over van getbootstrap.com en zet de eerste twee links minified CSS en optional theme in de header. Zet de laatste javascript include link aan het eind van de body onderaan de pagina.

Creëer dan de Bootstrap container:

<div class=”container”>

<div class=”row”>

<div class=”col-md-6 col-md-offset-3” id=”emailForm”>

<h1>My email form</h1>

<p class=”lead”>Uw email wordt zo snel mogelijk -

normaal binnen 24 uur – beantwoord” /p>

</div>

</div>

</div>

Dit creëert een Boostrap container van 6 kolommen met een offset van 3 kolommen, zodat onze 6 kolommen mooi in het midden liggen (van de 12 beschikbare kolommen).

De ruimte die we gebruiken geven we de klassenaam “emailForm”, die gestyled wordt:

<style>

#emailForm {

border: 1px solid grey;

border-radius:10px;

margin-top:20px;

}

</style>

Definieer nu het eerste (naam) veld van het formulier direct na de <h1> tag:

<form method=”post”>

<div class=”form-group”>

<label for=”name”>Uw naam:</label>

<input type=”text” name=”name” class=”form-control”

placeholder=”Uw naam” />

</div>

</form>

Doe dit vervolgens voor email, gebruik hiervoor dezelfde code en vervang “naam” door “email”. Zet for en type op “email”.

Vervolgens het comment veld. Dit wordt geen input maar een textarea:

<textarea class=”form-control” name=”comment”

id=”commentTextArea”></textarea>

Zet de hoogte van het comment veld op 120px:

.#commentTextArea {

height:120px;

}

Dan nog de submit button:

<input type=”submit” class=”btn btn-success btn-lg” name=”submit”

value=”Submit” />

Voeg nog wat padding onderaan het formulier toe om het af te maken:

form {

padding-bottom: 20px;

}

De javascript validatie van de invoervelden slaat Rob over, dit is al tijdens eerdere lessen aan de orde geweest. Rob verzorgt nog wel de serverside validatie, dit moet altijd gebeuren omdat javascript nog wel eens uit staat bij de client.

De terugkoppeling zet Rob direct onder de header. In de code voegt hij daarvor een klein stukje php code toe direct onder de <h1>:

<?php echo $result; ?>

De ‘echte’ php code zet Rob helemaal bovenaan de pagina, dus nog vóór de <!doctype html>. Allereerst de test of het formulier is verzonden (de submit knop is ingedrukt):

if ($\_POST[“submit”]) {

$result = “Formulier verzonden”; // test string

}

Omdat we Bootstrap gebruiken kunnen we de melding een stuk mooier maken:

$result = ‘<div class=”alert alert-success”>Formulier verzonden</div>’;

Let op dat je single quotes gebruikt voor de php string en double quotes voor de class, omgekeerd mag ook. Dezelfde quotes gaat echter niet, dan is voor de php interpreter niet meer duidelijk waar de php string eindigt.

Nu de check of de velden zijn ingevuld – ze moeten allemaal worden ingevuld:

if (!$\_POST[“name”]) {

$error = “<br />Naam veld mag niet leeg zijn”;

}

if (!$\_POST[“email”]) {

$error = “<br />Email adres veld mag niet leeg zijn”;

}

if (!$\_POST[“comments”]) {

$error = “<br />Opmerking veld mag niet leeg zijn”;

}

if (!$error) {

$result = ‘<div class=”alert alert-danger”><strong>Formulier bevat

fout(en):’ . $error . ‘</strong></div>’;

}

Dit werkt op zich echter wordt alleen de laatste fout getoond. Daarom de errors achter elkaar plakken:

$error .= “ foutmelding “;

De naam en comment veld worden verder niet gecheckt, maar het email adresveld wel. In javascript hebben we hiervoor een reguliere expressie gebruikt. PHP heeft hiervoor een handige oplossing voorhanden: de **filter\_var** functie. De info daarover is te vinden op de php.net site, in het manual. Dit is even zoeken, maar het staat er duidelijk met code voorbeelden erbij!

De email validatie wordt nu:

if ($\_POST[“email”]) {

if (!filter\_var($\_POST[“email”], FILTER\_VALIDATE\_EMAIL) {

$error .= “<br />Ingevoerd email adres is ongeldig”;

}

} else {

$error .= “<br />Email adres veld mag niet leeg zijn”;

}

De grap is nu dat Bootstrap al email validatie toevoegt op ‘email’ input velden, waardoor onze php validatie code niet wordt bereikt. Bij wijze van test veranderen we daarom even het type naar ‘text’.

Als alles in orde is dan kan het formulier via de mail verzonden worden. Hiervoor maken we een else clause bij de if ($error):

if ($error) {

// foutmelding

} else {

if (mail([casagrande2407@gmail.com](mailto:casagrande2407@gmail.com), “Comment from my website”,

“Name: “ . $\_POST[‘name’] .

“<br />Email: “ . $\_POST[‘email’] .

“<br />Comment: “ . $\_POST[‘comment’]) {

// succes message

$result = ‘<div class=”alert alert-success”><strong>

Thank you</strong></div>’;

} else {

$result = ‘<div class=”alert alert-danger”><strong>

Sorry uw email kon niet verzonden worden,

Probeer het aub opnieuw</strong></div>’;

}

}

Het is nu bijna klaar, het formulier werkt. Echter als één van de velden niet wordt ingevuld dan verschijnt bij het verversen van de pagina een leeg invoerveld en moet alles opnieuw ingevuld worden. Om dit te voorkomen zetten we de laatst ingevoerde data in elk invoer veld door een stukje php code toe te voegen. Bij het naam invoerveld gaat dit als volgt:

<div class=”form-group”>

<label for=”name”>Uw naam:</label>

<input type=”text” name=”name” class=”form-control”

placeholder=”Uw naam”

value=”<?php echo $\_POST[‘name’]; ?>” />

</div>

Dit werkt identiek voor email veld. Een <textarea> heeft geen value, daarom zetten we de php code daar rechtstreeks in, juist voor de closing </textarea>.

De email wordt nu verzonden. Omdat we het afzender veld leeg gelaten hebben kan het als spam beoordeeld worden, dus even checken of het wel aankomt. In de PHP documentatie staat echter dat een leeg afzender veld een foutmelding geeft.

**Les 187 – Project: Weather Scraper**

In deze les wordt een website gebouwd die weer informatie toont afkomstig van een bron op internet. De info wordt middels Ajax getoond, dus geen page refreshes nodig, en middels php uit de bron gehaald. De stad kan ingevoerd worden waarvoor de weersinfo wordt opgevraagd en getoond. Vrijwel alle tot nu toe opgedane kennis wordt toegepast: html, css, javascript, Bootstrap en php. De les bestaat uit een aantal challenges die uiteindelijk de weer website opleveren.

Gestart wordt met de bouw van een pagina die schermvullend is en een achtergrondfoto laat zien die Rob van **unsplash.com** haalt.

Rob begint met het definiëren van de container div:

<div class=”container”>

<div class=”row”>

<div class=”col-md-6 col-md-offset-3”>

</div>

</div>

</div>

Hij gebruikt dus 6 kolommen (van de 12 beschikbare) en met een offset van 3 kolommen staan deze in het midden van het scherm.

Vervolgens maakt hij de style aan voor het image. Hij styled de container omdat er maar één voor deze website wordt gebruikt, anders zou het via een class of id voor het background image moeten:

<style>

.container {

background-image:url(“images/seaview.jpg”);

}

</style>

Wanneer je de pagina laadt zie je nu echter niets. Zet vervolgens height en width van de .container op 100%, zodat de volledige ruimte met de foto wordt opgevuld. Geen verschil. Dit komt omdat de <html> en <body> geen hoogte hebben. Zet beide op 100%:

html, body {

height:100%;

}

Dit begint er al meer op te lijken. De foto is echter niet groot genoeg voor het hele scherm en wordt herhaald om het scherm te vullen. Dit lossen we op door

Image-repeat:no-repeat;

Aan de .container style toe te voegen. De foto wordt nu eenmalig in originele grootte getoond waardoor een groot deel van het scherm wit blijft. Dit lossen we op door

background-size:cover;

aan de .container style toe te voegen. De foto wordt nu geschaald getoond. Daarmee zijn we er nog niet, want wanneer de browser kleiner wordt gemaakt schaalt de foto maar deels mee tot hij zijn ware grootte bereikt, waarna het rechter deel van de foto wordt geclipped, de foto wordt niet gecentreerd weergegeven. Dit lossen we op door

background-position:center;

aan de .container style toe te voegen. Hiermee wordt altijd het midden van de foto weergegeven. Ziet er best mooi uit zo!

Dan de volgende challenge: toon de titel van de website centraal, met een beknopte handleiding eronder en daaronder een input veld waar een plaatsnaam ingevoerd kan worden.

Voeg de content in de binnenste (col-md-6) div toe:

<h1>Frank’s weerpagina</h1>

<p class=”lead”>Voer de naam van de stad in waarvan je de weersvoorspelling

wil zien</p>

<form>

<div class=”form-group”>

<input type=”text” class=”form-control” name=”city”

id=”city” />

</div>

<button type=”input” class=”btn btn-success btn-lg”

id=”zoekStad”>Zoeken</button>

</form>

Rob gebruikt geen label voor het input veld, ik heb dat wel gedaan. De button heb ik zelf niet gemaakt, is ook niet echt nodig, maar het ziet er zo wat uitnodigender uit.

NB Het button type is “submit”. Rob laat dat achterwege, maar als ik het goed begrijp is “submit” het default type van een button.

De tekstkleur verandert Rob naar wit en hij centreert de tekst. Hiervoor maakt hij twee klassen ‘center’ en ‘white’ aan:

.center {

text-align: center;

}

.white {

color:white;

}

De center class wordt aan de ‘col-md-6’ div toegevoegd. Bij Rob heeft dit echter geen effect, bij mij wel. Lijkt alsof er bij hem met de aanpassing iets mis is gegaan? Hij voegt de center class apart toe aan de <h1> en <p>. De white class wordt ook aan beide toegevoegd.

Het geheel aan tekst beweegt Rob wat naar beneden door 150px padding-top toe te voegen aan de container class. Dit werkt bij mij echter voor geen meter. Zelfs als ik de padding in de container op 0px zet, ontstaan er brede witte randen op de pagina. Ik heb daarom een style voor de <h1> aangemaakt:

h1 {

padding-top:150px;

}

Rob voegt wat meer tussenruimte toe aan de elementen door bij elk wat padding toe te voegen:

p {

padding-top:20px;

}

De button beweegt Rob naar beneden:

button {

margin-top:20px;

}

Ook voegt Rob nog een placeholder toe aan het inputveld. Ik wil daarvoor de laatst ingevoerde naam gebruiken ($\_GET[“stad”]), maar dit lijkt nog niet erg te werken. Komen we nog op terug…

De lay-out is nu in orde. Nu moeten de weersgegevens nog opgehaald worden. Dit doet Rob dmv ‘scraping’: gebruik de inhoud van een andere website, nl. **weather-forecast.com**. De reden dat Rob deze site gebruikt is dat de naam van de stad in de url voorkomt, waardoor we het eenvoudig kunnen manipuleren. Als de naam van de stad een spatie bevat, zoals San Franciso of New York, dan moet de spatie eruit gehaald worden voordat de url wordt gecreëerd.

De code zet Rob in een aparte php file **scraper.php**, deze zal later vanuit de challenge.php file worden aangeroepen.De volgende challenge is om de html inhoud van de opgevraagde pagina binnen te halen.

Rob gebruikt hiervoor de php functie **file\_get\_contents** die als argument de url meekrijgt. Het is dus echt een kwestie van de php site doorzoeken op mogelijkheden, deze functie is in de cursus nl. nog niet aan de orde gekomen:

echo file\_get\_contents(“http://www.weather-forecast.com/

location/San-Francisco/forecast/latest”);

Hiermee tover je dus de webpagina op je scherm! Het is echter niet de bedoeling dat we de pagina één op één overnemen. Rob is geïnteresseerd in de 3-daagse weersvoorspelling op de pagina. Om dit er uit te halen bekijkt Rob de sourcecode van de pagina. Met behulp van een reguliere expressie haalt Rob de gewenste informatie van de pagina, dit staat na een kopje ‘3 Day Weather Forecast Summary’:

$contents = file\_get\_contents(---url---);

Een reguliere expressie in php werkt met de functie **preg\_match**:

preg\_match(“/3 Day Weather Forecast Summary/i”, $contents, $matches);

print\_r($matches);

Het resultaat wordt in array $matches opgeslagen. Dit geeft als resultaat:

Array([0] => 3 Day Weather Forecast Summary)

Met de volgende techniek kun je het stuk tekst tussen twee opgegeven specifieke stukken tekst opvragen:

preg\_match(“/3 Day (.\*?) Summary/i”, $contents, $matches);

Dit geeft het volgende resultaat:

Array([0] => 3 Day Weather Forecast Summary [1] Weather Forecast)

Hiermee krijg je dus niet alleen de tekst die matched met de string terug, maar ook het deel tussen in dit geval “3 Day” en “Summary”. Om nu de gewenste tekst te krijgen moeten we deze afkaderen. In ons geval betreft het de tekst tussen ‘3 Day Weather Forecast Summary:<b><span class=”phrase”>’ en ‘</span>’:

preg\_match(‘/3 Day Weather Forecast Summary:<b><span class=”phrase”>

(.\*?)</span>/s’, $contents, $matches);

Let op dat de ‘/i’ vervangen moet worden door ‘/s’ omdat we een zg. multi-line check uitvoeren. Let ook op het gebruik van single quotes om de regexp string heen, omdat de string zelf een stuk met double quotes bevat (“phrase”).

Het resultaat bestaat zoals gemeld uit 2 delen, index 0 van $matches bevat de gehele string die matched met de vraag en index 1 bevat het deel tussen de opgegeven tekstfrasen.

De url moet nu nog de naam van de ingevoerde stad bevatten. Stel dat deze is opgeslagen in de variabele $city, dan wordt de url als volgt:

$contents = file\_get\_contents(“http://www.weather-forecast.com/

location/” .$city . “/forecast/latest”);

De spaties moeten nog uit de ingevoerde stadsnaam verwijderd worden. Dat gaat in php met de **str\_**replace functie als volgt:

str\_replace(“ “, “”, $city);

Het is de bedoeling dat de data dmv Ajax calls worden opgehaald. De code wordt nu in de challenge.php file aangeroepen. De code moet worden uitgevoerd als op de ‘Zoeken’ button is gedrukt, met id “#zoekStad”. Als eerste probeersel wordt het script in scraper.php aangeroepen met als parameter ‘city’ met waarde ‘London’:

$(“#zoekStad”).click(function(event) {

event.preventDefault();

$.get(“scraper.php?stad=London”, function(data) {

alert(data); // bij wijze van test

});

});

NB. Zet de code juist voor de </body> afsluiter.

Uitleg van de programmacode:

* event.preventDefault() is nodig om te voorkomen dat het formulier gesubmit wordt. Staat dit er niet dan wordt de $.get niet uitgevoerd.

**NB In eerste instantie dacht ik dat deze call overbodig was. Maar het bleek dat ik de <form> definitie vergeten was. Na toevoegen wordt scraper.php alleen aangeroepen als de event.preventDefault() wordt uitgevoerd. Het is dus zoals Rob zegt in de les.**

* data is de data die door het aanroepen van scraper.php wordt teruggegeven. In scraper.php staat als laatste statement ‘echo $matches[1]’, dit is feitelijk de output van scraper.php die in de ‘data’ parameter wordt gestopt.

In plaats van ‘London’ moet de ingevoerde stadsnaam worden doorgegeven aan scraper.php. Er moet ook gecheckt worden of er überhaupt een stadsnaam is ingevuld:

$(“#zoekStad”).click(function(event) {

// NIET NODIG: event.preventDefault();

if ($(“#stad”).val() != “”) {

$.get(“scraper.php?stad=” + $(“#stad”).val(), function(data) {

alert(data); // bij wijze van test

});

} else {

alert(“Vul aub een stadsnaam in”);

}

});

De weerinfo wordt nu in een alert weergegeven als een stadsnaam is ingevuld. Is de stadsnaam leeg dan verschijnt een fout alert.

In plaats van de huidige alert maakt Rob eenBootstrap succes alert die de info toont. Deze alert wordt onder de zoekknop getoond, maar alleen als er weer informatie beschikbaar is. Voeg ná de </form> de volgende div toe:

<div id=”success” class=”alert alert-success”></div>

De alert moet initieel niet zichtbaar zijn en een stukje onder de zoek button staan:

.alert { // zowel voor success als fail alert!

display: none;

margin-top: 30px;

}

Vervang nu de alert(data) als op de knop is gedrukt door:

$(“#success”).html(data).fadeIn();

Dit heeft tot gevolg dat de inhoud wordt voorzien van het via scraper.php ontvangen weerbericht en dat de alert via een fading animatie wordt getoond.

De website is bijna af. Het weer van de ingegeven stad wordt op een nette manier getoond. Als er een niet bekende/bestaande stad wordt ingevoerd dan verschijnt er een lege alert box. Dit gaan we vervangen door een foutmelding. Hiervoor maakt Rob een aparte ‘fail’ alert aan, zet deze direct na de success alert:

<div id=”fail” class=”alert alert-danger”>Stad niet gevonden. Probeer het

opnieuw</div>

$.get(“scraper.php?stad=” + $(“#stad”).val(), function(data) {

if (data != “”) {

$(“#success”).html(data).fadeIn();

} else {

$(“#fail”).fadeIn(); / LET OP: zonder html(data)

}

});

Een niet bestaande stad geeft een rode alert met de foutmelding. Deze blijft echter staan als daarna een bestaande stad wordt ingevoerd. Voeg een hide() call toe in beide gevallen:

$.get(“scraper.php?stad=” + $(“#stad”).val(), function(data) {

if (data != “”) {

$(“#fail”).hide();

$(“#success”).html(data).fadeIn();

} else {

$(“#success”).hide();

$(“#fail”).fadeIn(); / LET OP: zonder html(data)

}

});

De foutmelding die verschijnt als geen stad is ingevoerd is nu een lelijke alert box. Deze maken we overeenkomstig de andere alerts:

<div id=”noCity” class=”alert alert-danger”>Geen stad ingevuld. Probeer het

opnieuw</div>

En bij de else tak van de test op een lege stad:

if ($(“#stad”).val() != “”) {

// zoek weerinfo stad op

} else {

$(“.alert”).hide();

$(“#noCity”).fadeIn(); / LET OP: zonder html(data)

}

Voordat nu een alert wordt getoond worden eerst alle alerts uitgezet, dmv de $(“.alert”).hide() call. Dit is natuurlijk eenvoudiger dan alle alerts afzonderlijk verbergen.

**Section 8 – MySQL**

**Les 191 – What is MySQL?**

MySQL is een open source database waarmee je het geheugen van je webapplicatie kan realiseren. Open het control panel van je account bij EcoWebhosting. Ga naar de MySQL icon ne klik er op om een nieuwe database te creëren. Vul een usernaam en password in (ik gebruik casag2407 – mag max. 9 chars lang zijn - en een gegenereerd password tMgxFJj6z).

Onder de kop manage MySQL databases vind je jouw databases. Wordpress had er al één voor me aangemaakt. Klik nu op de Manage knop van de zojuist gecreëerde database. Er verschijnt nu een database management programma phpMyAdmin dat geschreven is in php. Klik nu op de database naam links in het scherm. De database is nog leeg.

Creëer nu een tabel met 5 kolommen. Vul de veldnamen ‘name’, ‘email’, ‘password’ in, dit zijn TEKST velden. Geef het vierde veld de naam ‘id’, stel het type in op BIGINT en maak dit de PRIMARY index. Vink de AutoIncrement A\_I vlag aan, zo krijgt elke nieuwe user een uniek id. Dit id veld wordt automatisch ingevuld. Laat het vijfde veld leeg, dit hebben we bij nader inzien nog niet nodig. Klik nu op Opslaan, het overzicht met de aanwezige tabellen verschijnt, in mijn geval alleen ‘users’.

Klik op de users tabel, er verschijnt een vrijwel leeg scherm omdat er geen users zijn gecreëerd. Creëer een nieuwe user door op de Insert (Invoegen) knop te drukken. Het leuke is dat de SQL instructie waarmee de user wordt aangemaakt ook zichtbaar is. Selecteer nu de Verkennen (Browse) tab, nu is te zien dat de users tabel 1 record bevat.

Rob kijkt verder nog rond naar de mogelijkheden die phpMyAdmin biedt, ziet er best indrukwekkend uit moet ik zeggen.

**Les 192 – Connecting to a database**

In deze les wordt behandeld hoe je vanuit je website een connectie met een database kunt maken.

Creëer op EcoWebhosting een directory mysql en creëer daarin een nieuwe file index.php. Voor het maken van een connectie gebruiken we de functie **mysqli\_connect()**. Dit is de verbeterde versie van mysql\_connect waarvan het gebruik afgeraden wordt. De call is als volgt:

mysqli\_connect(<host>, <user>, <password>, <database>);

echo mysqli\_connect\_error();

In mijn geval (zie mijn MySQL databases pagina op EcoWebhosting):

* <host> = “localhost” omdat mijn database op dezelfde server staat als waarvan dit programma draait. Het ip adres - 176.32.230.43 in mijn geval - kan ook gebruikt worden maar als dit verandert dan werkt het script niet meer.
* <user> = “cl58-casag2407”
* <password> = “tMgxFJj6z”
* <database> = “cl58-casag2407”

De functie **mysqli\_connect\_error**() geeft in geval van een fout de foutstring. Deze is leeg als de connectie zonder fouten geslaagd is. Het is natuurlijk beter om de foutdetails niet op de website te tonen aan derden – potentiële hackers – dus in geval van een fout een algemene foutmelding geven ‘cannot connect to database’ oid.

**Les 194 – Retrieving data**

Wanneer het niet mogelijk is om verbinding met de database te maken dan zal de website waarschijnlijk in zijn geheel niet werken. In plaats van een foutmelding te tonen is het dan ook beter de **die**() methode aan te roepen:

die(“Geen verbinding mogelijk met database”);

De foutmelding wordt getoond waarna de executie van het script stopt. Statements na de die() call worden niet meer uitgevoerd.

Om data uit een database te halen moet een sql query worden gedaan op de database. Dit gebeurt als volgt:

$link = mysqli\_connect(<host>, <user>, <password>, <database>);

if (mysqli\_connect\_error()) {

die(“Geen verbinding mogelijk met database. Probeer het later

nog een keer”);

}

$query = “SELECT \* FROM users;

if ($result = mysqli\_query($link, $query)) {

$row = mysqli\_fetch\_array($result);

print\_r($row);

} else {

“Opvragen van data mislukt”;

}

Het is gebruikelijk om in sql queries de sql keywords in hoofdletter te schrijven en de tabellen, velden etc in kleine letters.

De **mysqli\_query**() voert de query uit en geeft het resultaat ervan terug in een array in niet direct bruikbare vorm (een soort meta resultaat). Met **mysqli\_fetch\_array**() wordt het eerste record van het array resultaat opgehaald. Hoe we volgende records ophalen is nog niet duidelijk.

**Les 196 – Adding data to a database and updating data**

Data invoegen gaat ook dmv een sql query, maar nu met het INSERT statement:

$query = “INSERT INTO `users` (`name`, `email`) VALUES(‘Marcella”, “Brazilie”);

Let op dat voor tabel- en veldnaam **achteroverliggende quotes** gebruikt moeten worden (toest links van de Z). Voor de string moeten dan weer gewone quotes gebruikt worden.

Run je het script meerdere keren dan worden telkens nieuwe records met dezelfde gegevens aangemaakt. Elk record heeft echter een uniek id, wat we bij het creëren van de tabel opgegeven hebben.

Het updaten van gegevens gaat met het UPDATE statement:

$query = “UPDATE `users` SET `email`=’ian@hotmail.com’

WHERE `id`=3 LIMIT 1;

Als je de WHERE clause weglaat dan wordt het emailadres van alle records aangepast. De LIMIT 1 zorgt ervoor dat er maar één record wordt aangepast. Hier in feite overbodig omdat we met `id`=3 precies 1 record selecteren.

Let op: De where clause is niet case sensitive. Wil je bijv. alle records met name ‘frank’ updaten dan worden ook records met name=”Frank” meegenomen.

**Les 198 – Looping through data**

Tot nu toe hebben we met de SELECT query en de mysqli\_fetch\_array() 1 record van het resultaat getoond. In deze les wordt behandeld hoe je alle resultaat records kunt benaderen. Dit gaat met een while loop als volgt:

while ($row=my\_sqli\_fetch\_array($result)) {

print\_r($row);

}

NB Als je de paginabron in de browser toont dan ziet het resultaat – de opgevraagde records – er een stuk beter uit dan in de browser zelf. Het toont ook dat gegevens zowel met een numerieke index als met de veldnaam als index kunnen worden opgevraagd. Zo geven bijv. $row[1] en $row[‘email’] hetzelfde resultaat.

Andere voorbeelden van queries:

$query = “SELECT \* FROM users WHERE email LIKE ‘%h%’;

Hiermee worden alle records opgevraagd waarvan het emailadres veld een ‘h’ bevat. Het %-teken voor en na de ‘h’ geeft aan dat daar elke willekeurige rij karakters voor ingevuld kan worden.

$query = “SELECT \* FROM users WHERE name LIKE ‘b%’;

Hiermee worden alle records opgevraagd waarvan het naamveld met een ‘b’ begint.

NB Het LIKE commando is case-insensitive, de naam mag dus met een ‘b’ of een ‘B’ beginnen om geselecteerd te worden.

$query = “SELECT \* FROM users WHERE id<4 AND password!=’’”;

Hiermee worden alle records opgevraagd waarvan het id<4 is en het password ingevuld is. Let weer op de enkele en dubbele quotes.

Met de ‘\*’ worden alle velden van de opgevraagde tabel geselecteerd. Dit kan vertragend werken bij een flinke database. Het is ook mogelijk om specifieke velden op te vragen:

$query = “SELECT `name`, `email` FROM users WHERE id<4 AND password!=’’”;

Het aantal opgevraagde records kan als volgt bepaald worden:

echo mysqli\_num\_rows($result);

Een speciaal karakter, zoals de quote, kan als volgt ingebracht worden:

$query = “UPDATE `users` SET `name`=’Ian O\’Neil’ WHERE `name`=’Ian’”;

Het opvragen van records waarvan het name veld een quote bevat kan niet op de standaard manier:

$name = “Ian O’Neil”;

$query = SELECT `name` from users WHERE name=’”.$name.”’”;

Dit gaat niet werken omdat de inhoud van $name een single quote bevat. De manier om dit op te lossen is de functie **mysqli\_real\_escape\_string**() te gebruiken:

$query = SELECT `name` from users

WHERE name=’”.mysqli\_real\_escape\_string($link, $name).”’”;

**Les 200 – Session variables**

Een sessie variabele is een variabele die zijn waarde behoudt zolang de gebruiker actief is op de website. Bij een page reload houdt een sessie variabele zijn waarden. Pas wanneer de bezoeker de pagina sluit verliest de sessie variabele zijn waarde.

Om dit te gebruiken moet als **allereerste** statement een call naar **session\_start**() plaatsvinden, dus nog voordat er HTML code oid op de pagina gezet wordt. Vorbeeld:

<?php

session\_start();

$\_SESSION[‘loginid’] = 1;

echo $\_SESSION[‘loginid’];

?>

Run deze code en de waarde 1 verschijnt op een verder lele pagina. Haal nu de 2e regel ($\_SESSION[‘loginid’] = 1;) weg en reload de pagina. De output is hetzelfde. Dit lijkt op een static variabele uit C.

De waarde van een sessievariabele kan ook op andere webpagina’s gebruikt worden, zolang ze maar op dezelfde server staan. Maak een file test.php aan en zet daarin dezelfde code – dus enkel de session\_start() en de echo $\_SESSION[‘loginid’]. Run deze code en de waarde 1 wordt weer getoond.

**Les 202 – MySQL cookies**

Om gebruikers gedurende een langere periode ingelogd te laten blijven worden cookies gebruikt. Daar is wel toestemming voor nodig van de gebruiker. Voor een cookie is geen session\_start() aanroep nodig. Een cookie wordt als volgt aangemaakt:

setcookie(“id”, “1234”, time()+60\*60\*24);

Dit betreft een cookie met de naam ‘id’ en waarde ‘1234’ die vanaf het moment van instellen 1 dag geldig blijft, daarna vervalt hij.

De cookie kan nu vanaf andere php files benaderd worden.

Het verwijderen van een cookie gebeurt door de expiratie tijd in het verleden te zetten:

setcookie(“id”, “”, time()-3600);

De waarde van de cookie doet er niet meer toe. Om dit te effectueren moet de pagina ververst worden, anders blijft de cookie bestaan tot hij zijn expiratiemoment bereikt.

**Les 204 – Storing passwords securely**

In deze les worden 4 niveaus van password beveiling behandeld. Rob raadt aan om het hoogste niveau altijd te gebruiken.

Niveau 1 is het ongecodeerd opslaan van een leesbaar password in een tabel.

Niveau 2 is het gecodeerd opslaan van een password met een zg. hash. Hiervoor kun je de md5 functie gebruiken:

echo md5(“my password”);

Dit is een éénrichting vertaling, er komt een hashcode uit in de vorm van een enorme string uit bestaande uit cijfers en letters. Dit ziet er redelijk veilig uit, is dat echter niet. Op de site **crackstation.net** is van veelgebruikte passwords de hashcode opgeslagen. Je kunt daar een hashcode invoeren en kijken of er een match is met een daar bekend password. Zo ja dan is je password gehackt.

Op niveau 3 wordt gebruik gemaakt van een zg. salt waarmee het password wordt gecodeerd in de md5 functie. De ‘salt’ is een willekeurige string van cijfers en letters die je vóór je password zet in de md5() functie. Hiermee kunnen voor de hand liggende passwords zo gecodeerd worden (met bijv. md5()) dat ze nauwelijks nog te achterhalen zijn.:

$salt = “234jk23jhk23jkh4234jk”;

echo md5($salt . ”password”);

De salt moet uiteraard geheim blijven, anders ben je niets opgeschoten.

Op niveau 4 gebruik je een variabele salt, dus één die per gebruiker verschilt. In het voorbeeld gebruikt Rob het opgeslagen id van de gebruiker, deze wordt gehashed en het resultaat als salt gebruikt. Deze salt wordt samen met het password nogmaals gehashed voordat het in de account database wordt opgeslagen:

echo md5(md5($row[‘id’]) . ”my password”);

Dit is volgens Rob behoorlijk veilig. In de commentaren echter wordt gesteld dat md5 gedateerd is en makkelijk te kraken. Er wordt gewezen op **bcrypt** en **scrypt**.

**Les 206 – Project: Secret Diary (1)**

Het project van dit hoofdstuk is een webapp waar bezoeker een account kunnen aanmaken en kunnen inloggen om vervolgens een dagboek aan te leggen waar alleen zij toegang toe hebben. In hun dagboek kunnen ze zoveel entries aanmaken als ze willen.

Rob start met het bouwen van het login systeem op basis van een simpel formulier dat hij later netjes styled. Dit formulier bestaat uit een invoerveld voor emailadres, voor password en een submit button:

<form method=’post’> // niet vergeten!!

<input type=”email” name=”email” id=”email” />

<input type=”password” name=”password” id=”password” />

<input type=”submit” name=”submit” value=”Aanmelden” />

</form>

De volgende checks worden ingebouwd als de submit button is ingedrukt:

* is er een emailadres ingevoerd?
* Is het ingevoerde emailadres een geldig emailadres?
* Is het password ingevoerd?
* Is het password minstens 8 karakters lang?
* Bevat het password tenminste 1 hoofdletter?

<?php

if ($\_POST[‘submit’]) {

if (!$\_POST[“email”])

$error .= “<br />Voer een emailadres in”;

else

if (!filter\_var($\_POST[“email”],

FILTER\_VALIDATE\_EMAIL)

$error .=

“<br />Ingevoerd email adres is ongeldig”;

if (!$\_POST[‘password’])

$error .= “<br />Voer een password in”;

else {

if (strlen($\_POST[‘password’]) < 8)

$error .= “Password moet minstens 8

karakters lang zijn”;

if (!preg\_match(‘`[A-Z]`’, $\_POST[‘password’])

$error .= “Password moet minstens één

hoofdletter bevatten”;

}

if ($error)

echo “Fouten in aanmelding: “ . $error;

else {

// meld gebruiker aan.

}

}

?>

Het stuk ‘meld gebruiker aan wordt nu uitgewerkt. Voordat de gebruiker wordt toegevoegd aan de database wordt gecheckt of hij daar al in voorkomt. Daartoe moet een database connectie opgezet worden en het ingevoerde email adres opgezocht worden.

$link = mysqli\_connect(<host>, <user>, <password>, <database>);

if (mysqli\_connect\_error()) {

die(“Geen verbinding mogelijk met database. Probeer het later

nog een keer”);

}

$query = “SELECT \* FROM `users` WHERE

email=’”.mysqli\_real\_escape\_string($link, $\_POST[‘email’]).”’”;

$result = mysqli\_query($link, $query);

$results = mysqli\_num\_rows($result);

echo “Aantal gevonden records: “ . $results; // Bij wijze van test.

LET OP: de mysqli\_real\_escape\_string() call is nodig om SQL injection te voorkomen. Vul je nl. in het emailadres veld de tekens **‘);** in dan denkt de SQL interpreter dat het huidige commando beëidigd wordt en een nieuwe volgt. Vul je bijv. na deze tekenreeks het volgende in:

SELECT \* FROM `users`;

dan worden alle gebruikers uit de database opgehaald.

Het aantal gevonden records moet 0 zijn, als dat zo is kan de gebruiker toegevoegd worden:

if ($results) echo “Deze gebruikersnaam bestaat reeds, wil je inloggen?”;

else {

$query = “INSERT INTO `users` (`email`, `password`)

VALUES (‘”.mysqli\_real\_escape\_string($link, $\_POST[‘email’]).”’,

‘”.md5(md5($\_POST[‘email’]).$\_POST[‘password’]).”’)”;

mysqli\_query($link, $query);

echo “Je bent aangemeld!”;

}

**Les 207 – Project: Secret Diary (2)**

Om een gebruiker in te loggen gebruiken we een session variabele. Deze wordt op de server bewaard en blijft actief zolang het browser window van de gebruiker open blijft staan.

Zet aan het begin van het php script

session\_start();

En direct nadat de nieuwe gebruiker is aangemeld:

echo “Je bent aangemeld!”;

$\_SESSION[‘id’] = mysqli\_insert\_id($link);

Het interne id van de zojuist aangemaakte gebruiker wordt via de call naar mysqli\_insert\_id($link) opgehaald en toegekend aan de sessie variabele met de naam ‘id’. Hier kan elke willekeurige naam voor gekozen worden, Rob kiest voor ‘id’.

NB Het blijkt dat het id gelijk is aan het id dat automatisch door MySQL wordt gecreëerd als een nieuwe gebruiekr is aangemaakt. Dit is het nummer uit de ‘users’ tabel.

De server weet nu wie is aangelogd en kan hem redirecten naar de diary pagina. Daarvoor moeten echter eerst een aantal zaken geregeld worden. Ten eerste is het fijn als bij foute input deze in de invoervelden blijft staan:

<form method=’post’> // niet vergeten!!

<input type=”email” name=”email” id=”email”

value=”<?php echo addslashes($\_POST[‘email’]); ?>” />

<input type=”password” name=”password” id=”password”

value=”<?php echo addslashes($\_POST[‘password’]);” />

<input type=”submit” name=”submit” value=”Aanmelden” />

</form>

De functie **addslashes**() voegt backslashes toe aan vreemde karakters zoals double quotes, mochten deze voorkomen in email of password. Dit zou normaliter problemen geven met html code.

We hebben alleen een aanmeldformulier, er moet ook een inlogformulier komen. Dit lijkt sterk op het aanmeldformulier. Dit betekent tevens dat als op de submit button van één van beide formulieren wordt gedrukt we moeten weten op welke knop is gedrukt. Dit doen we door een verschillende waarde – resp. ‘Aanmelden’ en ‘Inloggen’ – aan de knoppen toe te wijzen. Het inlogformulier wordt nu:

<form method=’post’> // niet vergeten!!

<input type=”email” name=”loginemail” id=”loginemail”

value=”<?php echo addslashes($\_POST[‘loginemail’]); ?>” />

<input type=”password” name=”loginpassword” id=”loginpassword”

value=”<?php echo addslashes($\_POST[‘loginpassword’]);” />

<input type=”submit” name=”submit” value=”Inloggen” />

</form>

De php code voor het afhandelen van het ingevulde formulier geldt voor het aanmeldformulier en mag dus alleen worden uitgevoerd wanneer op de Aanmelden button is gedrukt:

$error = "";

if ($\_POST['submit'] == "Inloggen") {

if (!$\_POST['loginemail']) $error .= "<br />Vul aub een emailadres in";

if (!$\_POST['loginpassword']) $error .= "<br />Voer een password in";

if ($error) echo $error;

else {

echo "Connecting to database...<br />";

$link = mysqli\_connect("localhost", "cl58-casag2407",

"tMgxFJj6z", "cl58-casag2407");

$error = mysqli\_connect\_error();

if ($error) {

echo "Geen verbinding mogelijk met database";

} else {

echo "Succesvol verbinding gemaakt

met de database<br />";

$query = "SELECT \* FROM `users` WHERE

`email`='".mysqli\_real\_escape\_string($link, $\_POST['loginemail'])."'

AND `password`='".md5(md5($\_POST['loginemail']) . $\_POST['loginpassword'])."'";

$result = mysqli\_query($link, $query);

$row = mysqli\_fetch\_array($result);

$result = mysqli\_num\_rows($result);

if ($result) {

echo "Je bent ingelogd!";

// initialiseer sessie id (zie verder)

} else echo "Ongeldige combinatie van email en password.

Probeer het opnieuw";

}

}

}

Maar Rob had natuurlijk net zo goed de submit knoppen een verschillende naam kunnen geven. De check of het emailadres is gevonden doet Rob iets anders, omdat hij de row gebruikt om het id van de gebruiker op te halen. Dit kan nl. niet met de msqli\_insert\_id($link) omdat de id niet zojuist in de database is gezet, waar dit bij het aanmelden van de nieuwe gebruiker wel het geval was. De variabele $row bevat echter de benodigde data:

$row = msqli\_fetch\_array($result);

if ($row) {

// maak session id aan

$\_SESSION[‘id’] = $row[‘id’];

// Redirect gebruiker naar ingelogde pagina (🡪 volgt later)

}

Nu de code groter wordt besluit Rob deze te scheiden, de php code zet hij in aparte files. De code kan vervolgens ingevoegd worden dmv. de **include**() functie, hierdoor wordt de aanwezige php code direct uitgevoerd.

Hij zet alle php code in een file **login.php**. De code om een connectie met de database te maken – die nu in login.php staat – zet Rob in een aparte file **connection.php**, mocht databasenaam en/of password veranderen dan hoeft dat maar op één plek.

De volgende stap is de webpagina er beter uit te laten zien. Dat doet hij met Bootstrap en wel de code die gemaakt is in de challenge van hoofdstuk 5 Bootstrap, deze staat in de file ‘Landing page.html’.

Het voert te ver dit hier allemaal op te nemen. Alleen de bijzonderheden worden vermeld, om te beginnen met het tonen van een melding wanneer er bij het inloggen/aanmelden iets fout gaat. Rob doet dit dmv. een Bootstrap danger-alert. Deze voegt hij onder de <p> div met de korte beschrijving van de website toe. Hij doet dit met php code:

<?php

if ($error)

echo ‘<div> class=”alert alert-danger”>’

.addslashes($error).’</div>’;

else if ($success)

echo ‘<div> class=”alert alert-success”>’

.addslashes($error).’</div>’;

?>

Ik heb dit ietwat aangepast. Als de $error variabele een waarde heeft wordt een alert-danger getoond. Is dat niet het geval en heeft de $success variabele een waarde (als de gebruiker is aangemeld of ingelogd) dan wordt een alert-success getoond.

Nu wordt de applicatie gebouwd waar het om draait: de Secret Diary. Hiervoor creëert hij een file mainpage.php. Dit is de pagina die verschijnt als je aangemeld bent. Het menu rechtsboven moet nu veranderd worden,

Het html deel hiervan is op hoofdlijnen hetzelfde als van de aanmeld pagina, maar de login- en aanmeld formulieren hebben we niet meer nodig omdat de gebruiker al is aangemeld/ingelogd. In de navigation bar maakt Rob één enkele optie Uitloggen, dit betekent dat we de collapse button niet meer nodig hebben. Het wordt dan als volgt:

<div class=”navbar navbar-default navbar-fixed-top”>

<div class=”container”>

<div class=”navbar-header pull-left”>

<a class=”navbar-brand”>Secret Diary</a>

</div>

<div class=”pull-right”>

<ul class=”navbar-nav nav pull-right”>

<li><a href=””>Uitloggen</a></li>

</ul>

</div>

</div>

</div>

De pull-right is nodig om de Uitloggen optie naar rechts te halen, zoals gebruikelijk voor deze optie is. Om de navbar-brand op dezelfde hoogte als de Uitloggen optie te krijgen, moet de navbar-brand de ‘pull-left’ class krijgen.

De contentcontainer onder de topbar wordt leeggemaakt – het aanmeldformulier is niet nodig – en er komt een textarea in waar de dagboektekst in ingegeven kan worden:

<div class=”container content-container” id=”topContainer”>

<div class=”row”>

<div class=”col-md-6 col-md-offset-3” id=”topRow”>

<textarea class=”form-control”></textarea>

</div>

</div>

</div>

En een stukje javascript om de hoogte van het textarea zo hoog in te stellen dat het grootste deel van het scherm beschikbaar is om tekst in te voeren. Voeg aan de <script> sectie de volgende regel toe:

$(“textarea”).css(“height”, $(window).height()-120;

Omdat we maar één textarea op de website hebben krijgen alle textarea’s van deze site de ingestelde hoogte. De tekst die de gebruiker invoert wordt via Auto Update om de zoveel tijd opgeslagen, dit is een stuk netter dan telkens op een button te moeten drukken.

Allereerst moet er voor de tekst een nieuw veld ‘dagboektekst’ in de users tabel worden aangemaakt. Daarna maakt Rob een aparte php file aan die de dagboektekst update. Deze file wordt via Ajax aangeroepen. Eerst maakt Rob de update van het dagboektekst veld in orde in de ‘updatedagboektekst.php’ file;

<?php

session\_start();

include(“database\_connect.php”);

$query = “UPDATE `users` SET `dagboektekst`=

‘”.$\_POST[‘dagboektekst’]”’

WHERE ‘id’=’”.$\_SESSION[‘id’].”’ LIMIT 1”;

mysqli\_query($link, $query);

?>

Om snel te kunnen testen maakt Rob onder aan de php code een tijdelijke <form> aan:

<form method=”post”>

<input name=”dagboektekst” />

<input type=”button” />

</form>

Voor de trigger om een update uit te voeren gebruikt Rob een toetsaanslag in het textarea. Hiervoor kan de jQuery **keyup**() functie worden gebruikt:

$(“textarea”).keyup(function() {

alert(“textarea veranderd”); // bij wijze van test

}

Voor het updaten gebruikt Rob de **post**() functie, vervang de alert door het volgende statement:

$.post(“updatedagboektekst.php”, {dagboektekst:$(“textarea”).val()} );

Voor een productie systeem is het beter de ajax functie te gebruiken omdat daarmee ook foutafhandeling mogelijk is, bijv. wanneer de internetverbinding weg is gevallen of de database niet beschikbaar is etc etc.

De tekst wordt nu bijgewerkt maar niet weergegeven als de mainpage met het textarea wordt getoond. Hiervoor voegt Rob in de definitie van het <textarea> veld een stukje php code toe waarin een (nog te creëren) variabele dagboektekstveld wordt getoond:

<textarea class=”form-control”>

<?php echo $dagboektekstveld; ?>

</textarea>

Op zich zou de volledige code die nodig is om de dagboektekst data uit de database te halen hier in opgenomen kunnen worden, maar dan wordt het wat overzichtelijk. Rob kiest ervoor om dit helemaal bovenaan de pagina te doen – dus nog vóór de <html> tag:

<?php

session\_start();

include(“database\_connect.php”);

$query = “SELECT dagboektekst FROM `users`

WHERE `id`=’”.$\_SESSION[‘id’].”’ LIMIT 1”;

$result = mysqli\_query($link, $result);

$row = mysqli\_fetch\_array($result);

$dagboekTekstVeld = $row[‘dagboektekst’];

?>

Ik had vreemde problemen met het weergeven van de tekst, deze stond wel in de database maar de pagina bleef leeg… Op een gegeven momentwerkte het goed en ik weet niet wat er veranderd is, maar goed. Bovenstaande code is ok.

Nog een aantal zaken moeten in orde gemaakt worden. Zo moet de logout gemaakt worden en moet na de aanmelding/login de mainpage getoond worden. Rob start met de logout. Dit is relatief eenvoudig: de link in de href moet nog worden ingevuld, dit kan met een verwijzing naar de aanmeldpagina:

<li><a href=”aanmeldform.php?logout=1”>Uitloggen</a></li>

Rob neemt een ‘logout’ parameter mee die hij gebruikt bij het weergeven van de aanmeldpagina. Let er op dat dit een GET variabele is, die via $\_GET[‘logout’] kan worden opgevraagd. Dit doet hij in de aanmelden.php file, direct na de session\_start():

if ($\_GET[‘logout’]==1) {

session\_destroy(); // WERKT NIET!! Zie verder

$message = “Je bent uitgelogd. Een fijne dag verder!”;

}

**NB Dit werkt bij mij niet goed.** Wanneer alle wijzigingen hieronder zijn aangebracht dan blijkt dat bij uitloggen en opnieuw aanloggen de ‘id’ session variabele geen waarde heeft in mainpage.php. Dit komt volgens mij doordat de session\_destroy() ook de session ‘id’ variabele verwijderd. Deze wordt weliswaar weer aangemaakt als een gebruiker zich aanmeldt of inlogd maar dan is er nog geen sessie gecreëerd, dat gebeurt pas wanneer mainpage.php wordt aangeroepen. Het eerste statement is een session\_start() en waarschijnlijk is de sessie ‘id’ variabele dan leeg.

Mijn oplossing is om direct na de session\_destroy() een session\_start() toe te voegen en warempel dan werkt het goed.

Later voegt Rob een extra voorwaarde toe nl:

if ($\_GET[‘logout’]==1 && $\_SESSION[‘id’]) {

// code

}

Hiermee is mijn probleem ook direct opgelost. De sesssion\_destroy() vindt nu alleen nog maar plaats als er een gebruiker is ingelogd (sessie ‘id’ vaariabale is niet leeg). Heeft me de hele middag en avond gekost om dit op te lossen…

De $message variabele wordt in aanmeldform.php middels een Bootstrap success alert weergegeven. Doe dit direct onder de andere alerts:

if ($message) {

echo ‘<div class=”alert alert-success”>’ . addslashes($message). ‘</div>’;

}

De webapp is nu bijna klaar. Wat nog ontbreekt is dat de gebruiker na anmelden/inloggen wordt doorverwezen naar de mainpage. Hiervoor biedt php de **header**() functie. Deze voegen we toe in aanmelden.php direct nadat een gebruiker succesvol is aangemeld cq. ingelogd:

header(“Location:mainpage.php”);

Hierdoor wordt mainpage.php direct aangeroepen.

**Section 9: API’s**

**Les 211: What is an API?**

Deze slaan we voor het gemak over, er wordt niets interessants behandeld.

**Les 212: The Google Maps javascript API**

Hiermee kan een kaart op het scherm worden getoverd.

Ga hiervoor naar developers.google.com of zoek op Google Maps API. Het eerste wat moet gebeuren is een API key opvragen bij Google op de website.

De website van Google developers is inmiddels zo veranderd dat de les verouderd is. De website is duidelijk genoeg om het zelf te regelen, er staan diverse code voorbeelden.

Zie [**https://developers.google.com/products/**](https://developers.google.com/products/)**.**

**Les 214: Google Maps GeocodingAPI**

Geocoding betreft het omzetten van een adres in (lat, long) coördinaten en/of meer informatie over het adres opzoeken.

NB De titel van de les is ten onrechte GeoCoding, dit is iets heel anders!.

Reverse Geo Coding is het tegenovergestelde: een adres opzoeken aan de hand van een (lat, long) coördinaat.

Deze api moet enabled worden op de Google developer site. Het werkt in principe als volgt. Stel een url samen waarin een adres is opgenomen, vraag dr url op en je krijgt xml locatie gegevens terug. Dit kan met de jQuery ajax() functie. Rob maakt hiervan een voorbeeld aan de hand van een url die hij heeft samengesteld. Dit gaat echter niet zomaar werken omdat de opgevraagde url buiten het eigen domein ligt en dat gaat niet met ajax., op de request volgt dan een ‘request denied’ exception. Er zijn echter een paar uitzonderingen: het opvragen van JSON en XML data. Als de server waarvan de XML data vandaan komt jou daarvoor toestemming heeft gegeven dan gaat dit werken. Bij Google is dit het geval dus we kunnen verder.

NB Dit kan alleen met ajax omdat we hier het datatype (XML of JSON) kunnen opgegeven.

Rob maakt een voorbeeld waarmee hij xml output opvraagt rechtstreeks via een url. Het alternatief is om JSON data op te vragen – dit specificeer je in de url – en de gewenste info er uit te halen. Rob gaat vervolgens de xml-string parsen en zoekt op het gewenste type-element in de string om bijv. de postcode er uit te halen.

NB Vergeet niet jQuery te includen:

<script src=”//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js”>

</script>

Versienummer zal uiteraard verschillen, we zitten al een stuk verder dan 1.11.0.

De code wordt nu als volgt:

$.ajax({

type: “GET”, // POST kan natuurlijk ook

url: <https://maps.googleapis.com/maps/api/geocode/xml?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY>,

dataType:”xml”,

success: processXML // de functie die na ophalen data wordt

uitgevoerd

});

NB Vul bij YOUR\_API\_KEY de eigen van Google ontvangen API key in.

De echte logica zit in de eigen processXML() functie. Rob vraagt de <address\_component> velden uit de geretourneerde XML string op. Hij gebruikt daarvoor de **find()** functie – waarmee je kunt zoeken in een string – en de **each()** functie, waarmee je binnen een string alle occurences van in dit geval ‘address\_component’ kunt doorlopen:

function processXML(xml) {

$(xml).find(“address\_component”).each(function() {

if ($(this).find(“type”) .text()== “postal\_code”) {

alert($(this).find(“long\_name”).text())§;

}

}

}

Hiermee worden alle occurrences van “address\_component” doorlopen. Als er binnen zo’n element een “postal\_code” element voorkomt, dan wordt het “long\_name” element van de adress component getoond.

**Les 215 – Challenge 1 – Postcode Finder**

In deze challenge wordt een eenvoudige postcode zoeker gebouwd. In een website kun je een adres invullen en daar wordt vervolgens de bijbehorende postcode van getoond – als deze gevonden kon worden.

Rob gebruikt de weather app als basis. Hij gebruikt een foto met een stadsgezicht als achtergrond. De witte tekst komt voor hem dan niet goed uit. Hij maakt de tekstkleur daarom zwart. Dat komt ook niet uit waarop hij de achtergrondkleur van het invulformulier – de col-md-6 <div> - wit maakt. Hiervoor voegt hij een klasse ‘whiteBackground’ toe:

.whiteBackground {

background-color: white;

padding: 20px;

border-radius: 20px;

}

Voeg deze ook toe aan de ‘col-md-6’ <div>. Omdat we maar één invoerveld hebben, hebben we geen formulier nodig. Hiermee kan ook de ‘event.preventDefault()’ vervallen, deze voorkwam dat het invulformulier ge-submit werd maar dat is er dus niet meer.

Rob gebruikt de xml variant van de Google geocoder. De code wordt dan als volgt:

<script>

$(“#findMyPostcode”).click(function(event) {

$.ajax({

type: “GET”,

url:

“[https://maps.googleapis.com/maps/api/geocode/xml?address=” + encodeURIComponent($(‘#adres’).val()) + “&key=YOUR\_API\_KEY](https://maps.googleapis.com/maps/api/geocode/xml?address=)”,

dataType: “xml”,

success: processXML,

error: processXMLError // zie verder

});

});

</script>

‘findMyPostcode’ is de id van de button waarop moet worden geklikt. ‘adres’ is de naam van het invoerveld.

De url bevat nu de waarde van het ‘adres’ invoerveld. Om problemen met vreemde karakters – zoals ‘, “, \ etc - te voorkomen gebruikt Rob de functie **encodeURIComponent**().

De functie processXML wordt aangeroepen als de request succesvol is uitgevoerd. De ‘ajax’ functie kent nog veel meer parameters die zouden kunnen worden gebruikt, maar daar ziet Rob van af.

function processXML() {

$(xml).find(“address\_component”).each(function() {

if ($this).find(“type”).text() == “postcal\_code”) {

$(“#success”).html(“De postcode die je zoekt is: “ +

$(this).find(‘long\_name’).text().fadeIn();

}

}

}

De gevraagde postcode bevindt zich in het element ‘long\_name’. Let ook op het verschil tussen double en single quotes, dat doet Rob bewust mocht het adres double quote bevatten.

Vervolgens bouwt Rob nog wat foutafhandeling in, bijv. als er geen adres component wordt gevonden. Hij doet dit met een globale variabele **result** die initieel 0 is en op 1 wordt gezet als een postcode is gevonden. Is bij afronden van processXML de waarde van result nog 0, dan wordt de’fail’ <div> getoond. Wanneer de button wordt geklikt moeten allereerst alle alerts verborgen worden:

$(“alert”).hide();

Het kan nog erger fout gaan, bijv als de ajax functie een foutcode terug krijgt. Hiervoor kan in de ajax call de error functie parameter toegevoegd worden, de daar genoemde functie processXMLError wordt uitgevoerd als er een fout in de ajax call optreedt. Deze functie toont een ‘fail2’ div:

processXMLError() {

$(“#fail2”).fadeIn();

}

Waar Rob geen rekening mee houdt is dat er meerdere resultaten teruggegeven kunnen worden. Dit is een kwestie van een loop maken en ze allemaal tonen. Ik heb dit in mijn versie gedaan die met JSON werkt:

function processReceivedJSONGeocodeData(data) {

console.log("processing received JSON data...");

if (data.status == "OK") {

var resultaat="";

console.log("Aantal gevonden resultaten: " + data.results.length);

for (i=0; i<data.results.length; i++) {

console.log("Gevonden resultaat " + (i+1) + ": ");

console.log(data.results[i].formatted\_address);

resultaat += data.results[i].formatted\_address.toString();

for (j=0; j<data.results[i].address\_components.length;

j++) {

if (data.results[i].address\_components[j].types.

toString().search("postal\_code") > -1) {

console.log("Postcode = " + data.results[i].address\_components[j].long\_name.toString());

resultaat += ", postcode: " + data.results[i].address\_components[j].long\_name;

resultaat += "<br />";

}

}

}

$("#success").html(resultaat).fadeIn();

} else {

$("#fail").html("Fout bij verwerking adres: " +

data.status).fadeIn();

}

}

Op de Google developer site is de structuur van het JSON geocode object beschreven.

Een aardige suggestie die Rob nog doet is de kaart met de postcode te laten zien.

**Les 216 - Twitter API**

De Twitter api werkt met een authenticatie systeem, daarom is het hier beter om met php te werken. Het resultaat geeft de api terug in de vorm van een JSON object.

In deze les laat Rob zien hoe je een API key en secret API key aanmaakt en een token en secret token voor je Twitter app. Zie de video voor details of probeer het gewoon bij Twitter. Je hebt wel een Twitter account nodig!

**Les 217-220 – Twitter API application**

In deze lessen wordt een web app gebouwd waarmee Twitter berichten geselecteerd worden obv een filter (aantal keer bekeken). Hiervoor wordt gebruik gemaakt van de Twitter API en wordt JSON data binnengehaald en ontleed. Omdat ik niks heb met Twitter en al met de Google API JSON data heb binnengehaald, sla ik het over. Het materiaal schijnt behoorlijk achterhaald te zijn als ik de reacties op de lessen mag geloven.

**Section 10 – Mobile Apps**

**Les 223 – Creating mobile apps with HTML5**

In deze sectie worden een aantal mobile technologiën behandeld:

* jQuery mobile: een touch-optimized web framework, een laag boven op jQuery waarmee je gemakkelijk web apps kunt bouwen
* Apache Cordova: hiermee kun je apps die gebouwd zijn in html, css en javascript wrappen in een format dat naar de Appstore of Playstore kan worden ge-upload en er van kan worden gedownload. Het biedt je ook de mogelijkheid om native functies van mobiele apparaten te benaderen, zoals de camera of bewegingen

Deze worden gebruikt om een sketch (teken) programma te maken met de standaard functies die daarbij horen. Als extra functie wordt het scherm leeggemaakt als je je telefoon heen en weer schudt.

Een voordeel van op deze manier coderen is dat het resultaat overal kan worden uitgevoerd (single codebase) en dat het OS er niet toe doet.

Een nadeel is dat apps die gebaseerd zijn op html5 langzamer zijn dat native apps. Er zijn echter wel manieren om de snelheid op te krikken, maar zo snel als native code wordt het nooit.

**Les 224 – Introducing jQuery Mobile**

Bekijk de info op de jQuery Mobile website, vooral de demo’s zijn interessant. Met JQM kun je responsive websites bouwen, lijkt wel wat op Bootstrap maar is meer bedoeld voor mobiele apparaten.

Download de laatste versie 1.4.5 – Rob gebruikt 1.4.2. De download bevat ook de demo’s die on-line te zien zijn. Open de index.html in de Demo folder, dit is gelijk aan de on-line demo versie.

Rob gebruikt deze index.html om een appje te bouwen. Open de index.html in TextWrangler. Hierin wordt gerefereerd aan de \_assets/js/index.js file, deze bevat de code tbv de demo. Rob laat deze er in zitten, komt straks kennelijk van pas. Normaliter gooi je deze er uit.

Een mobiele app in jQuery bestaat uit een aantal pages, die in één html file kunnen worden opgenomen. Ons voorbeeld bevat er eentje.

Rob haalt het JQM logo er uit, vervangt het door zijn eigen tekst ‘My App’ en gooit het versienummer er ook uit. Het menu en de searchbox laat hij staan. De rest gooit hij weg tot aan de content div.

Vervolgens verkleint hij het aantal menu items. Alleen de Home, Introduction, Buttons, Button widget en Checkboxradio widget opties laat hij over. De footer laat hij staan, alleen de verwijzingen naar jQuery verwijdert hij.

Daarna is de searchbox aan de beurt. Rob laat zien hoe deze werkt. De searchbox bevat een lijst met items die de uitkomst van zoekresultaten zijn. Elk listitem bestaat uit een aantal strings die een zoekresultaat vormen. Tik je in de searchbox één van deze strings in, dan krijg je het bijbehorende zoekresultaat te zien. De meeste filter items gooit Rob weg.

Tot slot verwijdert Rob de melding dat ajax functies mogelijk niet werken. Hiervoor duikt hij in de index.js file in de \_assets folder. Hij zoekt in de code naar de melding en maakt commentaar van het stuk dat de melding creëert en weergeeft.

Wat overblijft vormt de basis van de app die in de volgende les wordt gebouwd.

**Les 225 – Web App Mini Project**

In deze les bouwt Rob een cursus app op basis van wat in de vorige les is overgebleven van de JQM demo website.

Als eerste pakt Rob de navigatie aan. In de demo app verwijzen de menu items naar JQM pagina’s. Rob houdt het op één index.html pagina waar alles in komt te staan en content dynamisch wordt geladen. Dit werkt sneller dan aparte pagina’s laden en is daarom prettiger voor de gebruiker.

Rob gebruikt jQuery om de links in het menu af te vangen en bijbehorende content te tonen. Hij past het menu aan en gooit een aantal opties er uit. Alleen de Home en Checkboxradio Widget optie laat hij staan. De tekst van de Checkboxradio Widget verandert hij in ‘Chapter 1’ en van de onderliggende opties in ‘Part 1’ en ‘Part 2’. De filter tekst van deze opties maakt hij leeg, evenals de hrefs, waar we later iets anders van maken. De ‘data-ajax’ opties haalt hij daar ook weg, omdat we zelf gaan controleren hoe dit werkt. Om ervoor te zorgen dat het menu item altijd ‘collapsed’ (uitgeklapt) wordt getoond voegt Rob data-collapsed=’false” toe aan het Chapter 1 menu item. NB Dit werkt bij mijn versie van de demo code net even anders, ik moest heel wat weghalen voordat deze optie zijn werk naar behoren deed.

Om deze twee menuopties af te vangen geeft Rob ze een class ‘contentUpdate’. Wanneer op één van beide links wordt geklikt dan vangen we dat met jQuery af en tonen onze eigen content. De javascript code is als volgt:

<script>

$(“.contentUpdate”).click(function() {

alert(“clicked”); // bij wijze van test

});

</script>

**LET OP:** de class ‘contentupdate’ moet aan het <a> element toegekend worden, NIET aan het <li> element, anders gaat het vervolg niet goed werken.

Rob merkt op dat er 300 ms vertraging zit tussen het moment van klikken en het tonen van de alert. Dit is niet veel maar geeft in de praktijk een trage indruk van de app. Om dit te voorkomen gebruikt Rob de **vclick**() functie ipv de click() functie. Bij hem werkt dit prima, maar bij mij wordt de alert niet meer getoond. Het blijkt dat de functie niet herkend wordt en dat de syntax anders tegenwoordig anders moet:

jQuery(“.contentUpdate”).on(“vclick”, function(event) {

alert(“clocked”);

});

Zie de JQM site voor meer info.

Als er op een menu optie wordt geklikt moet de content worden aangepast. Hiervoor creëert Rob een class “myContent” dat hij toevoegt aan het ‘main’ content element <div role=”main”…> net boven de <h1> tag. Vervang de alert(“clicked”) nu door:

$(“#myContent”).html(“new content”);

NB Rob maakt een id van myContent, ik doe het met een class. Maakt verder niet veel uit.

Helaas werkt dit ook niet bij mij, de oude content blijft gewoon staan… wellicht is ook hier iets veranderd in JQM. Heb van alles geprobeerd, een id, een class, op verschillende plekken, allemaal met hetzelfde resultaat: nada.

**NB** Later blijkt dat het wel werkt als je de code helemaal onder aan de pagina opneemt. Begrijpen doe ik het nog steeds niet, maar het werkt nu tenminste.

Vervolgens zoekt Rob uit op welk submenu item geklikt is. Dit kan door beide submenu items Part1 en Part2 een eigen id te geven en voor beide een click() functie te maken. Het punt hierbij is echter dat je dan de navigatie breekt. Stel dat je een nieuwe pagina opvraagt en je wil daarna terug naar de huidige pagina, dan werkt de back button niet. Een nettere manier is om voor beide submenu items de “href=” optie te gebruiken. Rob vult voor de hrefs resp. #1-1 en #1-2 in voor de submenu items. Doe je het zo, dan zie je het submenu id in de url-balk verschijnen. Dit betekent dat de Back knop weer werkt.

Rob gaat vervolgens content toevoegen voor beide submenu items. Hij voegt hiertoe een div met nieuwe class ‘hiddenContent’ en classes ‘1-1’ resp’ ‘1-2’ toe voor beide submenu items:

<div class=”hiddenContent 1-1”>

Test content Part 1

</div>

<div class=”hiddenContent 1-2”>

Test content Part 2

</div>

De code na de click wordt als volgt:

alert($(this).attr(“href”); // bij wijze van test

$(“#myContent”).html(“New content”);

De alert geeft netjes de id weer van het submenu item waar op is geklikt, maar inclusief het #-teken. Dit moet er af zodat het id overblijft (‘1-1’ resp. ‘1-2’). Hiervoor kan de substring() functie gebruikt worden:

alert($(this).attr(“href”).substring(1);

Hiermee wordt de substring vanaf positie 1 genomen, positie 0 valt dus af. Hiermee kunnen we de content van het geselecteerde submenu item ophalen en neerzetten in de ‘content’ div die eerder gedefinieerd is:

var submenuItemClass = $(this).attr(“href”).substring(1);

var newContent = $(“.” + submenuItemClass).html();

$(“#content”).html(newContent);

Nu alles buiten de JQM div’s staat blijkt het toch te werken.

Vervolgens pakt Rob de Home menu optie aan. Deze refreshed nu de complete pagina (door de href=”../.”) in plaats van alleen de content. Vervang deze door “#home”. Vergeet het #-teken niet! Anders wordt de ‘home’ pagina gezocht en die bestaat niet.

Maak een div aan voor de home content:

<div class=”hiddenContent home”>

// kopieer hier de content van de main div

</div>

En het werkt.

Dan merkt Rob op dat als je direct een sub pagina – zoals …..index.html#1-1 – opvraagt, dat niet lukt, je komt nu altijd eerst op de home pagina terecht. Er is nl. nog geen mechanisme gemaakt om dit te ondervangen. Om dit op te lossen moet je de ingetoetste url afvangen. Dat kan met **window.location.hash**. Het window.location attribuut geeft de volledige url, en window.location.hash geeft alleen de hash, dus #1-1 in ons geval. Is er geen hash ingevuld in de url dan geeft het hash attribuut een lege string.

De code om de ingevoerde subpagina te tonen is zojuist al gemaakt. Rob kiest ervoor om de niet te herhalen maar om een functie te bouwen:

function displayContent(hash) {

var submenuItemClass = hash.substring(1);

var newContent = $("."+submenuItemClass).html();

$(".myContent").html(newContent);

}

if (window.location.hash) {

displayContent(window.location.hash);

}

jQuery(".contentUpdate").on("click", function(event) {

displayContent($(this).attr("href"));

});

Als laatste voegt Rob content toe aan de subpagina’s. Voor Part1 voegt hij een Youtube filmpje van de cursus in. Ik heb een ander filmpje ingesloten. Kies op Youtube een filmpje en selecteer vervolgens ‘delen’ en ‘insluiten’ (share – embed). Voeg de code in de betreffende div en voila. Let nog op dat je de volledige url – dus incl. https:/www – gebruikt omdat de sourcecode lokaal via de browser wordt afgespeeld.

Rob speelt nog wat met de afmetingen van het iframe. Hij zet de width op 100%, zet er een <h1> tekst (Chapter 1 Part 1) onder en past de tekst die bij de home pagina hoort aan, daar komt de inleidende tekst te staan. Niets bijzonders verder.

Dan definieert hij nog een style voor de hiddenContent class, deze wordt nu onderaan de content weergegeven en dat is niet de bedoeling. Het merkwaardige is dat ik niets zie onderaan de pagina.

Tot slot past Rob de search optie aan. Ik heb dat er uit gesloopt. Alles wat Rob daar doet is de list items die er staan verwijderen en vervangen door de list items van het menu, compleet met alle bijbehorende klasse, gewoon één op één kopiëren. In de filtertext kun je nog tekst invullen om het zoeken naar bepaalde delen van de cursus te vergemakkelijken.

De web app is hiermee klaar. Om hem als native app op een mobiel apparaat te draaien hebben we een programma Apache Cordova nodig, dat gebeurt in de volgende les.

**Les 226 – Developer Tools and Node.js**

Eerst Cordova downloaden vanaf cordova.apache.org. Rob download de source.zip versie 3.4.0, tegenwoordig (maart 2016) zit Cordova op 6.x, een enorm verschil in 2 jaar tijd! De source.zip kan ik ook niet vinden, Cordova moet via de commandline geïnstalleerd worden en draait op Node.js, dat was bij Rob’s versie ook al het geval.

Node.js maakt gebruik van de terminal. Geef het volgende commando:

npm install –g cordova

NB npm is de package manager. Het blijkt dat je administratorrechten moet hebben, dan wordt het commando

sudo npm install –g cordova

Hierna ga je naar de directory waar je de app wilt bouwen:

cd ~/documents

mkdir apps

cd apps

mkdir helloworld

cd helloworld

Het ~ symbool leidt je naar je user home directory. Rob maakt dus een helloworld app die hij in de overeenkomstige subfolder van apps zet. Dan creëert hij de app met cordova:

cordova create appfiles com.fdg.helloworld HelloWorld

Hierbij is appfiles een subfolder van helloworld folder, com.fdg.helloworld de unieke identifier voor de app (net zoals dat in iOS moet) en HelloWorld de naam van de app.

Ga nu naar de appfiles directory. Hier moeten alle files, die nodig zijn voor het platform waar de app op gaat draaien, naar toe worden gedownload. Rob richt zich op Android en iOS:

cordova platforms add ios // download iOS library

cordova platforms add android // download Android library

Tot slot moeten de plugins die nodig zijn worden gedownload. Op de Cordova website staan er een heleboel, het hangt er vanaf wat je wil gaan maken. De device en console plugin zijn sowieso nodig. Voorbeeld:

cordova plugins add org.apache.cordova.device

cordova plugins add org.apache.cordova.console

Kijk nu in de appfiles folder, daar staat alles in wat is gedownload. De www folder is voor ons van belang, daar gaat onze code voor de app in terecht komen. De index.html file is al aanwezig, deze bevat de nodige ‘opstartcode’. Wanneer je deze in de browser draait dan verschijnt de melding ‘connecting to device’, deze blijft staan en dat is omdat de code echt op een device moet draaien.

Voor een Android app heb je de Android Developer Tools nodig. Deze zijn van de gelijknamige website te downloaden (zoek op met Google). Download de Android SDK voor Mac, hier zit alles in wat nodig is, inclusief Eclipse en een emulator. Eclipse heeft inmiddels plaatsgemaakt voor Android Studio en we werken (maart 2016) met Android 6. De download van Android Studio bevat ook de Android SDK tools, dus hiermee heb je een complete ontwikkelomgeving.

Rob doet nog een noodzakelijk stap, nl. het pad vastleggen waar de Android Developer Tools op de pc staan. Dit is nodig zodat Cordova ze kan vinden wanneer er een app gebouwd wordt.

Als eerste geeft hij de folder met de Android Developer Tools een andere- makkelijker – naam en slaat deze op in zijn Documents folder ipv op de desktop. Geef in de Terminal dan het volgende commando:

vi ~/.bash\_profile

Alle paden zijn in .bash\_profile opgeslagen, wij gaan hier met de VI editor een pad voor de Android Developer Tools aan toevoegen. Voeg daar de volgende regel aan toe:

export PATH=$PATH:~/Documents/Android/sdk/tools

Met ‘Android’ de zojuist gecreëerde folder. De subfolders daarvan bestaan al. Geef in de vi editor daarna het command **:wq**, hiermee sla je de file op en verlaat je de Vi.

Sluit de Terminal daarna af, start hem weer op en geef het Android commando. Als dit wat doet dan staat het pad goed.

**Les 227 – Creating an Android App**

Het laatste wat geïnstalleerd moet worden is **Apache Ant**. Dit is een library waarmee java applicaties gecompileerd, assembleerd, getest en gedraaid kunnen worden. Download de laatste versie zip archief in de Binary Distributions sectie. Pak de zipfile uit, geef de resulterende folder een makkelijke naam (bijv **ant**) en zet deze in de eerder gecreëerde Android folder. Het pad naar Ant moet nog ingesteld worden, voeg de volgende regel toe aan ~/.bash\_profile:

export PATH=$PATH:~/Documents/Android/ant/bin

Sluit de terminal, start deze opnieuw op en geef het commando

ant –version

Als het goed is gegaan dan wordt de versie van de geïnstalleerde ant getoond.

Rob voegt nu de android library toe, mar dat was in de vorige les al gedaan (cordova platforms add android). Vervolgens opent hij Eclipse om de Android app te ontwikkelen. Wat nu volgt is vanwege Android Studio waarschijnlijk obselete, maar wellicht nog nuttig.

Bij het openen van Eclipse krijgt hij een foutmelding dat de Android SDK niet kon worden gevonden op de aangegeven locatie. Deze staat dan ook verkeerd, moet staan op /users/<username>/Documents/Android/sdk.

Vervolgens creëert Rob een nieuw project in Eclipse, een Android Project from Existing Code. Na het selecteren hiervan verschijnt een scherm om de root directory van het Android project op te geven. Hier vult Rob in Documents/apps/helloworld/appfiles. Na selecteren van deze dir verschijnt een schermpje met daarin het Android project en de Cordova library. Beide zijn nodig voor de HelloWorld app.

Als dit gebeurt is runt Rob de app in de simulator, dit is een icoon boven aan het scherm met een wit pijltje dat omlaag wijst. Dit start de Android SDK manager. Hiermee gaan we een nieuw virtueel device creëren waar onze app op kan draaien. Vanuit het Tools menu selecteert Rob de ‘Manage AVDs…’ optie (AVD = Android Virtual Device). In het scherm dat dan verschijnt selecteert Rob de tab ‘Device Definitions’. Er verschijnt dan een scherm met een flinke lijst met devices waaruit er één geselecteerd kan worden. Rob kiest voor de Nexus4. Vervolgens drukt Rob op de ‘Create AVD’ knop. Er verschijnt dan een lijst met kenmerken van het AVD die eventueel aangepast kunnen worden. Rob adviseert om dat niet te doen maar indien nodig/gewenst alleen de naam aan te passen. Overigens vinkt hij de front en back camera niet aan, het geselecteerde device heeft dus geen camera. Na klikken op OK verschijnt een schermpje met een overzicht van de device kenmerken. Klik op OK en er verschijnt een lijst met devices, waar onze nieuwe nu ook bij staat. Rob klikt nu op ‘Start…’, er verschijnt een schermpje met een paar opties die Rob laat voor wat ze zijn en hij klikt dan op de Launch button. Daarna verschijnt het scherm van ons device in de emulator, met de gebruikelijke Android icons die je op een telefoon tegen komt. Rob klikt op het telefoon icon en direct valt op hoe langzaam dit werkt. Het is dus handiger om dit op een echt device te laten werken. Dit schijnt direct vanuit Eclipse te kunnen door een Android device via USB te koppelen.

Rob laat vervolgens zien hoe je de HelloWorld app in de emulator kunt runnen. Maar na wat foefjes houdt de video ineens op. Helaas pindakaas…. Anyway het is inmiddels obsolete, in de reacties is ook te zien dat men er niet goed uit komt en dankzij de Android Development cursus van Rob weet hoe met Android Studio dit aan de gang kan worden gekregen.

**Les 228 – Creating an iOS app**

Het ontwikkelen van een iOS app is een stuk eenvoudiger dan een Android app – tenminste in 2014. Hiervoor is maar één stuk software nodig: XCode. Download dit en je kunt aan de slag.

Eerder hebben we de iOS library voor Cordova al gedownload. Als dat is gebeurd ga dan naar de folder Documents/apps/helloworld/appfiles/platforms/ios. Hierin staat de helloword.xcodeproj file, open deze met XCode. Neem een kijkje in de code links in het scherm (de www folder), selecteer een device bovenaan het scherm en run vervolgens de code. De simulator laat nu het startscherm van de app zien en dat het ‘device ready’ is, dwz Cordova draait.

De index.html file kan nu in XCode aangepast worden.

**Les 229 – Project – Etch-A-Sketch**

In deze les wordt een teken app gebouwd, zowel voor iOS als Android. Hiervoor download Rob het sketch.js script. Hiermee kun je op eenvoudige wijze canvas-drawing functionaliteit in je app opnemen. Zie voor meer info en download de site intridea.gtihub.io/sketch.js.

Download sketch.js en sla de code op in een file genaamd sketch.js, in de folder Documents/apps/helloworld/appfile/www/js. Het mag ook ergens anders maar dit is de meest voor de hand liggende plek.

jQuery is ook nodig, download eventueel en sla jquery.min.js op in dezelfde folder.

Edit vervolgens de index.html file van de helloworld app, verwijder alles wat betrekking heeft op de demo en niet nodig is. Laat de include van cordova.js er in staan en voeg de volgende includes toe:

<script type=”tekst/javascript” src=”js/jquery.min.js”></script>

<script type=”tekst/javascript” src=”js/sketch.js”></script>

NB Rob voegt de js folder niet toe voor sketch.js, dat lijkt me niet in orde. Blijkt inderdaad ook iets later in de video.

Zet bij de <title> tag de naam ‘Etch-A-Sketch’. Kopieer de basis drawing code vanaf de sketch.js site naar onze index.html file.

De $(function()… is een constructie die we nog niet eerder gezien hebben. Het is een zg. self-invoking functie, deze wordt automatisch uitgevoerd als de pagina volledig geladen is, aldus Rob.

We hebben nu de basis voor onze teken app draaien. De volgende stap is het toevoegen van de icons met kleuren bovenaan het scherm. Rob kopieert simpelweg de code van de sketch.js site en gooit de code van de basis weg. Bij het runnen blijkt echter dat de styling van de icons niet goed is. Daarvoor moet de css code uit sketch.js overgenomen worden in de index.html file. Dit gebeurt dus kennelijk niet wanneer de file wordt geladen in de <script> tag. Na het overnemen van de css code ziet het er in elk geval prima uit.

Omdat we de app op devices met een klein scherm willen laten draaien, moeten we de styling wat aanpassen. Rob verdeelt de iconen over een top deel (kleurtjes) en een bottom deel (lijndiktes en download button). Nu volgen de nodige acties om de styling in orde te krijgen. De iconen komen in 2 rijen direct onder elkaar te staan, de download knop haalt Rob eerst naar buiten en later verwijdert hij hem omdat onbekend is hoe dat uitpakt op een mobiel device.

In paats van de download knop zet Rob de Marker en Erase knoppen in onze app. De code daarvoor kopieert hij van de sketch.js site, het id zet hij om van ‘tools\_sketch” naar “colors\_sketch” omdat al het id van het canvas is.

Dan voegt hij nog een erase knop toe, om de tekening leeg te maken. Dit doet roep door de href leeg te maken, wat inhoudt dat de pagina ververst wordt, met als resultaat dat het scherm weer blanco is.

Vervolgens maakt Rob het gedeelte waarin getekend kan worden groter in vertikale zin. Dit doet hij met een stukje javascript:

document.getElementById(“colors\_sketch”).height=$(window).height()-100;

De breedte zet hij op de volledige breedte van het scherm:

document.getElementById(“colors\_sketch”).width=$(window).width();

De app is nu klaar en kan in de iOS simulator gedraaid worden, door simpelweg de .xcodeproj file te openen. Het valt wel op hoe traag dit bij Rob werkt.

Rob begint nog over de Android simulator maar dat komt er niet meer van.