design your future

Web Development 1 – Labo HTML 1

handelswetenschappen en bedrijfskunde

bachelor in de toegepaste informatica

campus Kortrijk

academiejaar 2021-2022

vives katholieke hogeschool
associatie KU Leuven

Inhoudsopgave

1 Inleiding

Dit labo is het eerste labo over HTML en omvat de eerste drie hoofdstukken uit de HTML5 cursus, met uitzondering van een paar stukjes die later aan bod komen. Dit document gaat dieper in op enkele topics uit deze drie hoofdstukken, alsook een paar bijkomende onderwerpen.

Tijdens de les wordt ook getoond hoe je Webstorm kunt gebruiken.

1.1 Verslag

In dit deel van de cursus staan verschillende vragen die je moet beantwoorden en opdrachten om iets te maken of uit te proberen. Het is belangrijk dat je alle opdrachten zorgvuldig uitvoert!

Documenteer je werk in een verslag document "**verslag HTML deel 1.pdf**" waarin je:

- Voor elke uitprobeeropdracht een entry maakt met screenshots ter staving van wat je deed.
- Je antwoorden op de gestelde vragen neerschrijft.

Oplossingen van 'grotere' opdrachten (met veel code) bewaar je in een Webstorm project "**HTML deel 1**". Per opdracht maak je in dit project een aparte folder waarin je de bestanden (en subfolders) plaatst.

2 Labo

2.1 HTML, CSS en Javascript

Lees de inleiding uit de gedrukte HTML-cursus. Deze inleiding bevat een belangrijk stuk waarin uitgelegd wordt dat moderne webontwikkeling drie verschillende technologieen gebruikt bij het definieren van een webpagina:

- HTML: om de inhoud en structuur vast te leggen.
- CSS: om de layout en visualisatie van verschillende elementen te bepalen.
- Javascript: om gedrag en dynamiek toe te voegen aan de pagina.

We zullen deze opsplitsing gedurende de ganse cursus toepassen en verwachten dat jouw oplossingen dat ook doen.

Lees vervolgens Hoofdstuk 1 over de geschiedenis van de HTML-standaard. Voor de opdrachten in dit labo is dit stuk niet meteen nodig, dus je kunt dit overslaan en thuis lezen.

2.2 HTML Document

Lees Hoofdstuk 2 uit de HTML-cursus. Het gedeelte over de basisstructuur van een HTML5 document is belangrijk, je zult dit voor veel opdrachten nodig hebben als startpunt.

Je kan trouwens in Chrome de broncode van het HTML document door rechts te klikken op de pagina en dan "View page source" te kiezen (of korter: op CTRL+U te drukken).

2.3 Opdracht 1

Surf naar https://www.HTMLdog.com/examples/headings1.html en bekijk de broncode. Vergelijk deze met de response body uit de originele HTTP-request voor deze pagina op het “Netwerk” tabblad van de Chrome Developer tools (het netwerk tabblad en HTTP requests kwamen in het vorige labo aan bod).

2.4 Opdracht 2

Maak een nieuw project aan in Webstorm en neem de basisstructuur van een HTML5 webpagina over in een nieuw HTML bestand. Bekijk dit vervolgens ook in Chrome.

2.5 HTML Elementen

Een HTML-document is dus een tekstdocument, waarin onderdelen van een tekst afgebakend worden met speciale markeringen die ook wel tags heten. Zo'n afgebakend onderdeel noemen we een “HTML element”.

Een markering begint met een begintag (van de vorm `<...>`, bv. `<p>`) en eindigt met een eindtag (van de vorm `</...>`, bv. `</p>`). Merk op dat begin- en eindtags op elkaar afgestemd zijn (de `...` is bij beiden gelijk, bv. `p`).

Lees nu Hoofdstuk 3 uit de HTML cursus, maar sla voorlopig Secties 3.4.4 t.e.m. 3.6 over. Vergeet niet om op het einde ook Sectie 3.7 over hyperlinks en afbeeldingen te lezen.

De meeste HTML-elementen hebben zowel een begin- en een eindtag. Sommige elementen (bv. `<img>`) behoeven echter geen eindtag omdat alle nodige informatie al in de attributen vervat zit (bv. de url van de afbeelding).

Een kleine uitbreiding van wat in het begin van Hoofdstuk 3 staat: de typische structuur van een HTML element is eigenlijk:

```
<element attribute="value">content</element>
```

waarbij `content` een combinatie kan zijn van tekst of andere elementen (nesting).

Bijvoorbeeld:

```
<p>Dit is een eenvoudige paragraaf</p>

<a href="www.example.com">dit is een link naar www.example.com</a>

<p>Dit is een eenvoudige paragraaf <a href="www.example.com">met een hyperlink</a> en”nog wat meer tekst</p>
```

In bovenstaande voorbeelden werd de content van een element onderstreept (let op de dubbele lijnen!). In het derde voorbeeld bevat het `<p>` element dus een `<a>` element.

Qua terminologie zeggen we dat het `<a>` element genest is in het `<p>` element en deze mogelijkheid in HTML noemen we ‘nesting’ (van het Engelse werkwoord “to nest”).

2.6 Opdracht 3

Plaats telkens een van de bovenstaande drie voorbeeldelementen in het body element van ons basisdocument, bewaar en bekijk telkens de pagina in Chrome.

Denk eraan op refresh te klikken, de browser weet immers niet dat het document op schijf is veranderd.

De browser negeert trouwens extra whitespace (lege ruimte zoals spaties of tab en newline karakters), zowel tussen woorden op eenzelfde regel als over meerdere regels. Het heeft dus geen zin om te proberen de layout te beïnvloeden met extra spaties of extra lege regels in het HTML bestand. Dit lijkt een nadeel maar is voor ons al programmeurs een voordeel omdat het ons toelaat de HTML code te schikken op een manier die ze voor ons duidelijk(er) maakt in een editor, zonder dat dit gevolgen heeft voor de gebruiker die deze in de browser bekijkt.

Je kan een HTML bestand dat op je harde schijf staat trouwens op twee manieren bekijken in je browser:

- via het `file://` protocol

    - door erop te dubbelklikken in de Windows verkenner
- via het `http://` protocol op de host 'localhost' op poort 63342
    - door in Webstorm 'Open in browser' te kiezen

Wat is het verschil tussen beide?

2.7 Opdracht 4

Plaats in een HTML document wat extra spaties tussen 2 woorden in een paragraaf en ga na dat dit daadwerkelijk geen invloed heeft op de pagina voorstelling in de browser. Plaats daarna enkele lege regels tussen 2 woorden en vergewis je ervan dat ook dit geen effect heeft op de visualisatie.

Je kan trouwens extra whitespace forceren door ` ` en `<br />` te gebruiken voor resp. extra spaties en nieuwe regels.

Gebruik deze echter niet als gemakzuchtige layout trucs om de ruimte tussen pagina onderdelen te corrigeren; daarvoor gebruiken we immers CSS!

2.8 Opdracht 5

Maak een HTML-document waarin de volgende tekst staat:

*Gebruik   en
 echter niet voor layout doeleinden, daarvoor gebruiken we immers CSS!*

Je zult hiervoor enkele entity references uit Sectie 3.1 moeten gebruiken.

2.9 Geneste elementen

“Correct genest” betekent dat de volgende eigenschappen gelden als we de tekst overlopen van begin tot einde:

1. Elke eindtag die we tegenkomen *moet* passen bij de begintag die we het laatst tegenkwamen en waarvoor we nog geen eindtag hadden gevonden.
2. Er *moet* voor elke begintag een passende eindtag in de tekst voorkomen (met enkele uitzonderingen, zie gedrukte cursus).

We kennen het principe van begin- en eindtags om stukjes tekst af te bakenen eigenlijk al uit de wiskunde: haakjes! Bijvoorbeeld: (3+4)x(5-2)

2.10 Opdracht 6

Waarom zijn de volgende uitdrukkingen niet correct genest?

- `(4x2 + 2(`

- `(4x2( +2`
- `(4x(2+2)`
- `(4x(2+2()`

Waar we in de wiskunde meestal maar 1 soort tag gebruiken (namelijk ronde haakjes) zijn er in de HTML taal verschillende soorten begin- en eindtags mogelijk: elk soort element heeft een eigen begin- en eindtag!

2.11 Opdracht 7

Veronderstel dat we in een tekst verschillende soorten haakjes mogen mengen: (), {} en []. De voorwaarde is echter dat ze correct genest moeten zijn. Zijn de haakjes in de volgende teksten correct genest?

- `({}[])`
- `{{{()]`
- `({()})`
- `(){[}]`
- `{{([])`

2.12 Opdracht 8

Veronderstel een HTML document met daarin, o.a.,

```
<span>hier komt een</span><a href="">link</a><span> te staan</span>
```

(Terzijde: is een spatie net voor het woord "te" eigenlijk nodig? Probeer eens uit!)

De auteur beslist plots dat de tekst "een link" benadrukt moet worden dmv. een `<strong>` element en verandert bovenstaande naar:

```
<span>hier komt <strong>een</span><a href="">link</a></strong><span> te staan</span>
```

Waarom is dit niet correct, en hoe moet het dan wel?

2.13 Opdracht 9

Reproduceer de volgende pagina (incl. afbeelding):

http://www.HTMLdog.com/examples/headings1.html

en gebruik hierbij onze HTML5 basis structuur. Bewaar dit als “voorbeeld headings.HTML” en open dit in de browser. Ziet het er correct uit?

Plaats de afbeelding in een folder ‘images’ in je project. Ziet het er nog steeds correct uit? Heb je de waarde van het `src` attribuut van het `<img>` element moeten aanpassen?

2.14 Opdracht 10

Wat is er verkeerd in onderstaand HTML fragment?

```
<ul>
<li>een</li>
<li>twee</li>
<ol>
<li>jan</li>
<li>piet</li>
<li>joris</li>
</ol>
<li>vier</li>
</ul>
```

2.15 Website of webapplicatie?

Soms is het wat verwarrend of we een bepaalde verzameling HTML documenten die we op het internet tegenkomen een website dan wel een webapplicatie moeten noemen. Er is beslist een grijze zone tussen beiden, maar misschien dat onderstaande hun focus wat verduidelijkt.

2.16 Website

Een website is een verzameling documenten die aan elkaar gekoppeld zijn doordat ze naar elkaar verwijzen binnen een bepaalde context (wat dat ook moge zijn, een domain naam, een sub domain, een url path, enz). Een website biedt vooral informatie aan en de interactie met deze informatie is eerder beperkt.

We onderscheiden:

- **statische websites**
  gebaseerd op fysieke HTML-bestanden, die op de webserver op schijf staan
- **dynamische websites**
  gebaseerd op HTML-documenten die per request gegenereerd worden door de server bv. het resultaat van een zoekopdracht op amazon.com (bekijk in chrome beslist eens de HTTP requests die bij je zoekopdracht hoorden!)

2.17 Webapplicatie

Webapplicaties bieden functionaliteit die boven het aanbieden van informatie uitstijgt en kunnen zeer ingewikkeld gedrag vertonen, bv. een commerciele site die rekening houdt met je interesses en/of een winkelwagentje bevat.

De gegevens die een webapplicatie moet bewaren (bv. taal voorkeur, inhoud winkelwagentje, ...) kunnen trouwens in de browser en/of op de server opgeslaan worden. De opslaglocatie die de

makers van de webapplicatie kiezen, hangt af van wat voor soort data het is en wat men wil bereiken.

Moderne browsers bieden verschillende opslagmogelijkheden waarvan een website bouwer gebruik kan maken: webSQL, cookies, localstorage of sessionstorage. Je kunt hun inhoud trouwens inspecteren op het Application tabblad in de Chrome Developer Tools, probeer dit eens uit op je favoriete site!

Om veiligheidsredenen worden belangrijke gegevens natuurlijk best aan de server kant bewaard en niet in de browser. Immers, data die door de browser wordt bewaard kan relatief eenvoudig achterhaald worden door eenieder die toegang heeft tot de lokale computer (privacy) of onbeveiligd aangepast worden door een gebruiker met slechte bedoelingen (tampering). Bv. als de totaalprijs voor een online bestelling enkel in je browser zou worden opgeslagen, zou je deze zelf kunnen aanpassen om minder te moeten betalen dus dat zou niet erg slim zijn van de makers van die webapplicatie.

2.18 Opdracht 11

We zullen eens kijken hoe de website van Microsoft informatie bijhoudt in de browser. Surf naar:

https://www.microsoft.com

Open nu de Chrome Developer Tools en kijk op het Application tabblad bij 'local storage' en 'cookies' welke data deze website door de browser laat bewaren. Je zult zien dat er per domein waar de webpagina gebruik van maakt, aparte data wordt bijgehouden. Wijzig eens de regio/taal van de pagina (linksonderaan de webpagina van microsoft) en kijk opnieuw naar de cookies. Zie je waar de gekozen taalinstelling wordt bijgehouden?

Opdracht 12

Surf naar: https://www.lg.com. Ga naar de televisie sectie. Daar kun je een overzicht bekomen van al hun actuele toestellen die je ook kan vergelijken. Je zult zien dat je verschillende modellen kunt vergelijken door ze aan te vinken en dan op de Compare knop te klikken. Merk op dat je gerust modellen op verschillende pagina's kunt aanvinken om toe te voegen aan je vergelijking.

Ergens moet deze webapplicatie dus bijhouden (over meerdere pagina's heen) wat we aangevinkt hebben. Open de Chrome Developer Tools en kijk op het Application tabblad bij 'local storage' en

'cookies' welke data daar bewaard wordt. Vink een aantal toestellen aan en kijk wat er verandert. Om de veranderingen te zien zul je wellicht op die 'local storage' en 'cookies' tabbladen op de kleine refresh knop moeten klikken (dus niet de grote refresh knop naast je adresbalk, die laadt immers de ganse pagina opnieuw in!).

Waar precies bewaart de webapplicatie welke toestellen je aanvinkte ter vergelijking? Ben je nog andere herkenbare informatie tegengekomen op die beide tabbladen? Zoja, welke?

2.19 Opdracht 13

We zullen eens kijken hoe men bij Amazon informatie bijhoudt over je winkelkarretje. Surf naar https://www.amazon.com.

Mocht je daar een account hebben en ingelogd zijn, gelieve uit te loggen voor deze opdracht.

Plaats een product in je winkelwagen en kijk opnieuw bij local storage en cookies. Wordt de inhoud van je winkelwagen daar ergens bewaard? Zoniet, waar dan wel denk je?

Als je het tabblad sluit en op een nieuw tabblad weer naar Amazon.com surft, zul je zien dat je product nog steeds in het winkelkarretje zit. Hoe weten ze bij Amazon eigenlijk welke winkelwagen bij jou hoort? Je bent immers niet ingelogd op hun site!