design your future

Web Development 1 – Labo CSS 3

Yanic Inghelbrecht, Christophe De Waele

handelswetenschappen en bedrijfskunde

bachelor in de toegepaste informatica

campus Kortrijk

academiejaar 2021-2022

vives katholieke hogeschool associatie KU Leuven

Inhoudsopgave

1 Inleiding

Deze les introduceert specificiteit en behandelt de opmaak van tabellen.

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 (pdf of docx) “**verslag CSS deel 3**” 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 "**verslag CSS deel 3**". Per opdracht maak je in dit project een aparte folder waarin je de bestanden (en subfolders) plaatst.

2 Labo

2.1 CSS hulpmiddelen in de Chrome developer tools

Een nogal grove aanpak om conflicterende regels op te sporen is stukken in het CSS-bestand in of uit commentaar te zetten en telkens het resultaat te bekijken, op die manier kun je proberen de problematische regel(s) op te sporen. Een meer verfijnde manier is echter de beschikbare hulpmiddelen in de browser te gebruiken!

In de Chrome developer tools kun je veel informatie vinden over hoe je CSS-regels worden toegepast door de browser. Als je een probleem hebt met CSS-regels niet toegepast te worden, of geen effect lijken te hebben, selecteer dan het Elements tabblad en kijk rechts op het Styles tabblad. De relevante regels staan (van boven naar onder) gerangschikt volgens dalende prioriteit. Daar kun je niet alleen zien wat de browser met je CSS-regels uitspookt, maar ook on-the-fly wijzigingen aanbrengen om te zien of dit het probleem oplost. Meer uitleg over de mogelijkheden vind je hier[1]. Wat meer uitleg over de betekenis van doorstreepte[2] en grijze[3] properties vind je op Stackoverflow.

Op het Computed Styles tabblad kun je nagaan door welke regel een bepaalde property wordt overridden (overschreven)[4].

2.2 Opdracht 1

Lees de documenten die hierboven worden worden aangehaald (zie voetnoten) en experimenteer met de aangehaalde technieken op verschillende pagina's. Bekijk ook telkens de relevante CSS-regels op een aantal elementen op je favoriete websites.

Snel en efficiënt kunnen werken met de hulpmiddelen in Chrome developer tools is erg belangrijk, stop de nodige tijd in het vertrouwd raken ermee!

2.3 Opdracht “Nature blog”

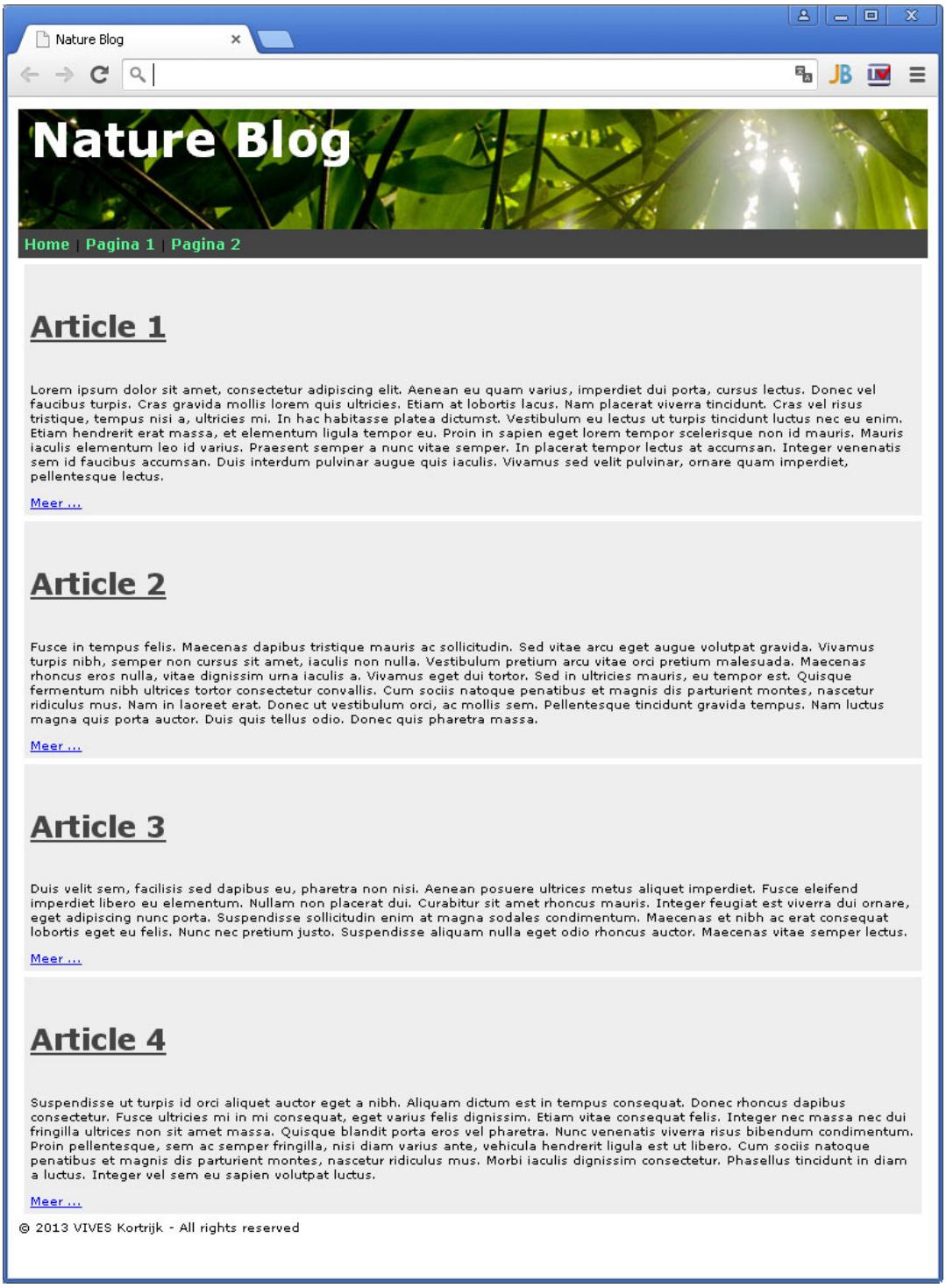
In de zipfile van deze opdracht vind je een HTML-file met de hoofdpagina van een blog en een afbeelding voor een header. Probeer de pagina in **Fout! Verwijzingsbron niet gevonden.** zo goed

[1] https://developer.chrome.com/devtools/docs/elements-styles

[2] http://stackoverflow.com/questions/3047056

[3] http://stackoverflow.com/questions/3265555

[4] http://stackoverflow.com/questions/13867088



Figuur 1 Visualisatie Opdracht "Nature Blog"

mogelijk na te bouwen. Zorg ervoor dat je pagina exact 1200px breed is. De resterende ruimte aan de zijkanten blijft leeg, je kunt dit klaarspelen door de margin op 'auto' te zetten van dat 1200px brede element.

2.4 Opdracht Persoonlijke homepagina met CSS

Herwerk je persoonlijke pagina (de oplossing waarin alles op één pagina staat) zodat er verwezen wordt naar een '`styles.CSS`' bestand met CSS style rules. Dit bestand moet in een '`styles`' subfolder van je project staan.

Maak de pagina mooier door 'zoveel mogelijk' zaken uit dit en het vorige labo toe te passen[5]. Kijk in de Chrome Developer tools op het Styles tabblad om te zien hoe met de stijlregels wordt omgegaan. Als een bepaalde style rule geen effect lijkt te hebben kun je in de Developer tools nagaan of de browser de regel eigenlijk wel toepast op het gewenste element. Indien er syntaxfouten in de regel zitten kun je dit merken aan een geel waarschuwingsicoontje.

Je kan op het Styles tabblad ook makkelijk iets uitproberen door de rules te editeren (via dubbelklik), dit is soms sneller dan telkens het CSS-bestand te bewerken. Uiteindelijk moet de definitieve versie natuurlijk wel in het CSS-bestand terechtkomen. Kopieer dan je tijdelijke aanpassingen uit de Developer Tools naar je eigenlijke CSS-bestand.

Ga beslist ook eens na hoe het met collapsed margins zit, de `<li>` elementen uit je lijsten zijn een goeie plek om daarmee te experimenteren. Je zult zien dat de oranje margin highlights elkaar dan verticaal kunnen overlappen.

2.5 Browser compatibiliteit

We zagen in een vorige les dat browsers nieuwe experimentele CSS-properties introduceren m.b.v. browser-prefixes. Op die manier worden een aantal potentiële toekomstige problemen vermeden, e.g., als de toegelaten waarden nog zouden wijzigen bij het finaliseren van de volgende CSS-versie.

Op onderstaande site kun je makkelijk terugvinden welke ondersteuning de verschillende browsers (en hun versies) bieden voor een bepaalde mogelijkheid in HTML, CSS of Javascript: http://www.caniuse.com.

2.6 Specificiteit

Lees Sectie 3.4.5 over de specificiteit van selectoren. De precieze regels moeten natuurlijk geen parate kennis zijn, maar je moet wel de principes kennen.

Om makkelijk de specificiteit van een selector te berekenen kun je bv. volgende 'specificity calculator' gebruiken: http://specificity.keegan.st/.

Soms kun je een regel meer voorrang t.o.v. een conflicterende regel geven door de specificiteit te verhogen zonder de betekenis te wijzigen. Bv. als je de class 'belangrijk' enkel gebruikt voor `<p>` elementen, dan zal de selector '`p.belangrijk`' een hogere specificiteit opleveren dan de selector '`.belangrijk`' maar nog steeds dezelfde elementen selecteren.

[5] Vermijd een overdaad aan *kitsch.*

Merk echter op dat specificiteit slechts 1 van de factoren is die de *prioriteit* van een CSS regel beïnvloeden. Vaak kun je prioriteitsproblemen ook oplossen door de regels in een andere volgorde in hun CSS bestand te plaatsen.

2.6.1 Voorbeeld

We maken een oplossing voor de 'checklist met checkmarks' opdracht uit de vorige les, op basis van de `::before` pseudoselector:

```
<ul class="checklist">
  <li class="checked">boter</li>
  <li>kaas</li>
  <li class="checked">eieren</li>
</ul>

...

.checklist li::before {      /* specificity = 012 */
  content: "\2610";          /* hokje zonder vinkje */
}

.checked::before {           /* specificity = 011 */
  content: "\2611";          /* hokje met vinkje */
}
```

De idee achter deze oplossing is goed: alle list items krijgen een leeg hokje via `::before` content maar voor de elementen met class `.checked` vervangen we dit door een vinkje. Dit zou in theorie kunnen werken omdat er maar 1x content via `::before` kan worden toegevoegd per element.

Helaas werkt deze oplossing niet, geen enkel element zal een vinkje krijgen omdat de eerste CSS-regel een hogere specificiteit heeft waardoor de tweede CSS-regel deze niet kan overrulen. De onderlinge volgorde is alleen relevant indien ze eenzelfde specificiteit hebben, dus de regels van plaats wisselen helpt hier niet.

Kopieer beide selectoren naar de online specificity calculator van hierboven en ga na dat de eerste regel inderdaad een hogere score krijgt.

We moeten er dus voor zorgen dat de selector in de tweede CSS-regel eenzelfde (of hogere) specificiteit heeft maar toch dezelfde elementen selecteert (let op de `li` vooraan de selector!):

```
.checklist li::before{ /* specificity = 012 */
  content: "\2610";    /* no checkmark */
}

li.checked::before {   /* specificity = 012 */
  content: "\2611";    /* checkmark */
}
```

De toevoeging van '`li`' in die selector heeft in ons voorbeeld geen impact op welke elementen geselecteerd worden (zowel `.checked` als `li.checked` selecteren hier dezelfde elementen), maar het levert wel een hogere specificiteit op.

Nu hebben beide regels dezelfde specificiteit en dan wint de regel die lager staat in de CSS-file. Deze oplossing werkt dus.

Mocht je de score van de tweede regel nog verder willen verhogen kun je bv. dit doen:

```
.checklist li.checked::before {   /* specificity = 022 */
  content: "\2611";               /* checkmark */
}
```

Dit zal nog steeds dezelfde `li` elementen selecteren als voorheen en nu kan je de beide regels in eender welke volgorde in je CSS-file plaatsen.

2.7 CSS tabellenopmaak

2.7.1 Opdracht 2

Maak een HTML tabel met persoonsgegevens van een vijftal imaginaire vrienden met minstens 6 kolommen (voornaam, familienaam, postcode, etc.).

Voeg deze tabel toe aan je persoonlijke pagina in een sectie 'Vrienden'.

Experimenteer met de instellingen op https://tablestyler.com/ en kijk naar de gegenereerde code op de HTML en CSS tabbladen. Ga na welke HTML elementen daar gebruikt worden en welke CSS properties aan deze elementen werden gekoppeld.

Helaas is de gegenereerde tekst niet netjes geformatteerd. Je kunt deze echter overkopieren naar Webstorm en daar mooi formatteren. Om terug te keren naar de tabel preview moet je nogmaals op het tabblad klikken.

Kies een ontwerp dat je mooi vindt en voeg de nodige code toe aan je persoonlijke pagina om je vriendentabel volgens het gekozen ontwerp weer te geven. Laat de `div` met `class="datagrid"` uit de generator achterwege (je zult dus de CSS-selectoren ietwat moeten aanpassen).

Opgepast! Het is de bedoeling dat je in deze opdracht met de generator experimenteert om te zien welk effect verschillende properties hebben op tabel onderdelen. Gewoon eentje kiezen en rap rap overnemen zal je weinig bijbrengen.

Beantwoord de volgende vragen:

- Zijn alle properties die de generator voorziet werkelijk nodig? Probeer dit uit.
- De generator gebruikt voor bepaalde properties een `browser` prefix. Zijn die voor jouw browser (nog steeds) nodig?

2.7.2 Opdracht 3

De generator geeft de `<tr>` elementen van elke even rij een '`alt`' class:

```
<tr class='alt'>...</tr>
```

De generator voorziet de cellen in deze rijen van een andere achtergrondkleur, zoek op waar dit precies staat in de gegenereerde CSS.

Voorzie in je oplossingen van de vorige opdracht enkele cellen met onbekende gegevens (bv. een geboortedatum die je niet kent, of niet zeker van bent). Zorg ervoor dat je zowel op een even als een oneven rij zo'n cellen hebt. Geef deze cellen een class 'onbekend' en stop een regel in je CSS waardoor ze een rode achtergrondkleur krijgen.

Waarom heeft de volgende CSS regel:

```
.onbekend {background-color:red;}
```

geen effect als je deze toepast op een `<td>` cel in een `<tr>` rij met `class="alt"`, maar wel in een `<tr>` rij zonder deze '`alt`' class? Hint: het heeft te maken met de specificiteit van de regels.

2.7.3 Opdracht Kalender

Maak een HTML-pagina met de tabel uit Figuur 2. Je ziet dat dagen uit een vorige/volgende maand grijs getoond worden. Weekends krijgen een roze achtergrond `rgb(255, 220, 220)`. Probeer dit eens zonder een speciale 'weekend' CSS-class te regelen. Voorbeeld met de `:first-child` en `:last-child` pseudo-class selectoren.

De 16e is een vrije dag en krijgt achtergrond `rgb(250,120,120)`. De maand hoofding heeft achtergrond `rgb(170,170,200)`. De dag hoofding krijgt tekstkleur `rgb(200,190,240)` en achtergrond `rgb(35,37,70)`.

June 2020						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3

Figuur 2 Illustratie Opdracht Kalender