> *A light went off in my head.* Responsive *felt right for what I was trying to describe: layouts that would just* know *the best way to fit on a user's screen. A user wouldn't have to tap or click on anything to get the best design for their laptop or smartphone; rather, the design could fluidly adapt to the space available. It'd just* respond.
> —Ethan Marcotte

CSS Responsive Web Design

1 Inleiding

Dit topic is waarschijnlijk het belangrijkst in heel dit OPO. Eén van de basisuitgangspunten is dat we ontwerpen voor iedereen, onafhankelijk van handicap (toegankelijkheid), toestel, besturingssysteem, enz. Het is onze plicht als front-end developer om zoveel mogelijk mensen toe te laten een site / app zonder moeilijkheden te gebruiken. Iemand legde in dit verband de term 'RWD' uit als 'Responsible (verantwoordelijk) Web Design'.

2 Steve Jobs 2007

In 2007 stelt Steve Jobs de eerste iPhone voor. De webdesign wereld zou nooit meer dezelfde zijn.

De iPhone was niet de eerste smartphone, maar het was een gemakkelijk te gebruiken ding. Essentieel in de visie van Jobs was dat er een degelijke browser aanwezig was: Safari.

Kort na deze voorstelling waren we met een aantal collega's web van UCLL (toen nog KHLeuven) op een tweedaagse webdesign conferentie in Londen. Twee boeiende dagen met 7 sprekers per dag die je meenamen in boeiende nieuwe evoluties. Er ging geen lezing voorbij of de spreker maakte een verwijzing naar de iPhone en hoe dit toestel een impact ging hebben op front-end development, wat nadien ook gebleken is.

3 Ethan Marcotte 2010

In 2010 schreef Ethan Marcotte voor A List Apart een invloedrijk artikel waarin hij een nieuwe term lanceerde: “Responsive Web Design”. In dit artikel gebruikt hij drie technieken om responsief te werken:

- Werk zoveel mogelijk flexibel, dus eerder met `%` dan vast (met `px`).
- Zorg dat afbeeldingen de volledige breedte van hun container vullen m.b.v. `max-width: 100%`
- Gebruik `@media` queries om bij bepaalde schermafmetingen CSS aan te passen.

3.1 Flexibel werken

Vroeger – voor RWD – werden ontwerpen meestal in pixels gemaakt. Dikwijls was de pagina 960 pixels groot (zie bvb https://960.gs, gek dat dit nog bestaat ...). Dat was logisch want de meest courante schermafmeting was 1024 px. Het getal 960 heeft superveel delers en kon dus gebruikt worden voor 3, 4, 5, 6, 8, 12, ... kolommen. En toen kwamen smartphones.

Een deel van de oplossing ligt in het afstappen van de px als meest gebruikte eenheid. Werken met procenten (`%`) of 'fraction' eenheden bij grids (`fr`) maakt dat een ontwerp zich gemakkelijker kan aanpassen aan de beschikbare grootte.

3.2 Afbeeldingen zich laten schalen

Om een afbeelding zich in beperkte mate te laten aanpassen aan de container waarin ze zit, kan je volgende heel eenvoudige regel gebruiken:

```
img {
  max-width: 100%;
}
```

De `img` zal nu automatisch kleiner worden als de container waarin ze zich bevindt ook verkleint. De maximale grootte van de afbeelding is wel beperkt tot het aantal pixels breedte van de afbeelding zelf. Een foto van 600px breedte, zal in een container van 800px toch maar 600px breed zijn.

Snap je dat dit toch eigenlijk wel een probleem is voor afbeeldingen die erg klein worden? Je laadt eerst een afbeelding met heel veel pixels, en dan wordt die heel verkleind weergegeven in de browser. Dat is niet erg efficiënt. Er bestaat een nieuwe manier om hiermee om te gaan en dat is gebruik maken van 'echte responsieve afbeeldingen'. Wie een uitdaging zoekt voor de eigen site: graag! Het behoort niet tot de standaardtopics voor dit OPO, maar ik zou het wel fijn vinden als je dit toch probeert te implementeren in jouw site.

3.3 @media queries

Een `@media` query is vergelijkbaar met een `if` uit sommige andere programmeertalen. Een klein voorbeeld:

```
@media (min-width: 30em) {
  body {
    font-size: 1.1rem;
  }
}
```

Deze regel zegt: “als de viewport minstens 30em breed is, dan mag je de CSS-regels in dit blok toepassen”. Mogelijk overschrijven deze CSS-regels waarden die erboven al gedefinieerd waren. In dit geval zal voor grotere schermen (vanaf 30em) de lettergrootte aangepast worden naar `1.1rem`.

> Tip: eerst definieer je in je CSS altijd de ‘gewone’ regels. Daaronder komen de ‘speciale gevallen’ m.b.v. een media query. Als je dus – zoals ik zelf probeer toe te passen – ‘mobile first’ werkt, dan staat bovenaan in je CSS alles voor een klein scherm (wat in principe veel gemakkelijker om te lay-outen is dan een groot scherm). Onder deze CSS-code staan dan mogelijk meerdere @media queries die steeds specialer worden. In dit geval dus: voor steeds grotere viewports.

4 Nieuwe technieken

CSS is niet blijven stilstaan sinds 2010. CSS staat trouwens nooit stil … We hebben ondertussen heel wat meer mogelijkheden die toen nog niet bestonden: flex, grid, CSS functies zoals `min()`, `max()`, `clamp()` enz.

5 Responsive Design Made Easy

> In volgend filmpje van 42 minuten geeft Kevin Powell een mooie demo i.v.m. responsief werken. Hij vertrekt van een tekening (gemaakt in een programma zoals Figma) van het eindresultaat, bedenkt dan de HTML (structuur) en vult tenslotte stapsgewijs de CSS in.

6 Meta viewport

Iets heel kleins, maar erg belangrijk. Zorg ervoor dat volgende regel in de `head` van je HTML-bestand staat. Deze regel zegt aan de browser van een smartphone “gelieve niet te liegen over je breedte, maar geef de echte breedte van het scherm zodat de site zich hieraan kan aanpassen”. Toen de eerste iPhone uitkwam in 2007 waren de meeste sites ontworpen voor een breedte van 960 pixels. Apple liet dat toestel liegen over zijn breedte en deed alsof de breedte 980 pixels was, zodat de meeste sites van toen netjes op het scherm pasten. Gebruikers konden daarna zoomen om de tekst toch op het klein scherm te kunnen lezen.

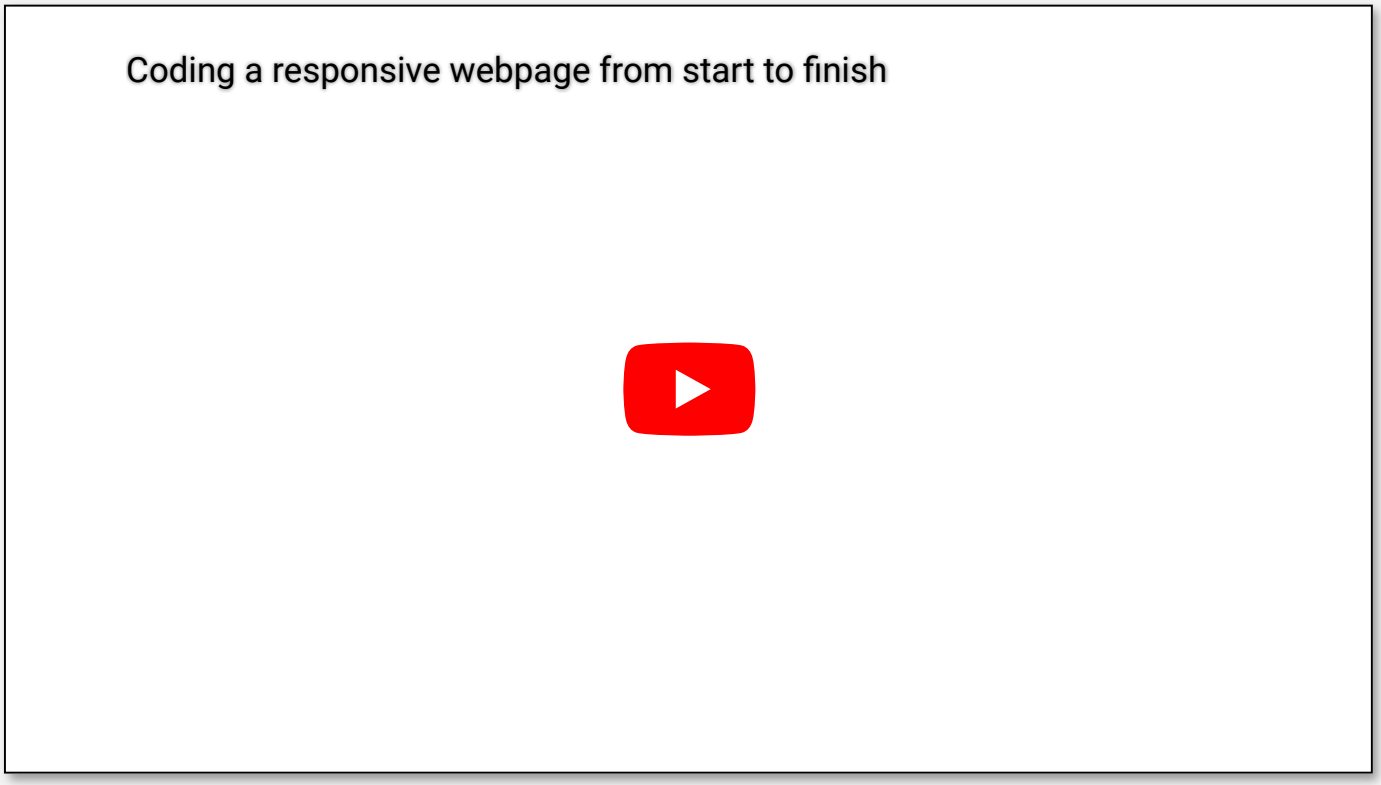
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Als je deze regel vergeet en test in de ‘responsive design view’ van de developer tools van Firefox, zal het daar lijken alsof je site zich netjes aanpast aan alle breedtes van smartphones. Op een echte smartphone krijg je echter de desktopversie te zien.

7 Responsive page from start to finish

Studenten vragen vaak “Hoe begin je aan zoiets?”. Hopelijk heb je wel al door dat je een site niet maakt door Visual Studio Code op te starten. Belangrijker is om eerst een *plan* te maken. *Een blad papier en een potlood zijn een ideaal startpunt. Schets!* Teken een versie voor klein scherm. Wat kan er gebeuren als je meer plaats ter beschikking hebt op een groter scherm? Denk dan na over de grote blokken en hoe je ze gaat positioneren en opdelen in kleinere structuren.

Als uitsmijter van dit topic nog een laatste keer Kevin, met een langere film (twee uur dit keer). Vertrekkend van een tekening codeert hij volledig *mobile first* de pagina, met redelijk wat gebruik van de `flex` eigenschap.



↑ Naar top