# 1. Static vs Adaptive vs Responsive

*“You must be shapeless, formless, like water. When you pour water in a cup, it becomes the cup. When you pour water in a bottle, it becomes the bottle. When you pour water in a teapot, it becomes the teapot. Water can drip and it can crash. Become like water my friend.” (Bruce Lee)*

Kemungkinan semua orang yang belajar CSS menunggu untuk belajar bagian ini karena sekarang semuanya mobile and PC dengan resolusi yang beragam. Bagaimana cara supaya CSS kita bisa masuk ke dimensi apa saja dan user agent apa saja?

Sebelum belajar cara membuatnya kita harus lebih dulu mengerti sifat-sifat tampilan, web design biasanya tergolong ke dalam 3 bentuk, yaitu:

**Static**

Static adalah web design yang tidak berubah tampilan dan struktur contentnya, di monitor sebesar apa pun atau di device apa pun. Pada saat memasuki layar yang lebih besar, web static akan memperluas backgroundnya atau marginnya. Pada saat memasuki layar yang lebih kecil, static biasanya akan membuat seluruh bagian webnya menjadi scrollable secara horizontal atau besar tampilannya akan diperkecil secara otomatis sesuai dengan besar medianya.

**Flexible/Liquid**

Lalu setelah besar monitor pada pc dan notebook semakin beragam, keluarlah konsep flexible. Yaitu mencoba memanfaatkan layar pc semaximal mungkin dengan menggunakan value-value dalam persentase dan inline, sehingga tampilan css bisa menyesuaikan dengan besar layar browser yang dipakai user.

**Adaptive**

Adaptive adalah web design yang strukturenya bisa dipaksa berubah menjadi apapun dengan menggunakan media query, yaitu fungsi css untuk membaca dimensi dan resolusi browsernya dan meng-override selector stylenya. Untuk lebih jelasnya mengenai media query akan dibahas pada chapter berikutnya.

**Responsive**

Responsive bisa dibilang merupakan gabungan dari flexible dan adaptive. Responsive adalah cara membuat tampilan yang paling baik dan yang paling disukai di jaman now.

Catatan: untuk meningkatkan performance pada responsive, developer minta semaximal mungkin untuk memanfaatkan unsur flexible ketimbang adaptive. Tetapi tetap jangan ragu untuk memanfaatkan sifat adaptive apabila memang dibutuhkan.

# 2. Media Query

Media query adalah cara css untuk menciptakan selector yang satu tingkat lebih kuat lagi, pada saat browser anda mencapai kondisi tertentu. Untuk pelajaran saat ini, yang akan saya kondisi yang akan saya fokuskan adalah viewport.

Apa itu viewport? Viewport adalah besar resolusi layar pada browser yang sedang digunakan. Ingat viewport itu belum tentu sama dengan resolusi monitor atau setting pada operating system.

Untuk lebih tau pasti viewport yang saat ini anda lihat pada browser anda, bisa menggunakan javascript dengan fungsi windows objectnya, atau bisa dengan beberapa plug-in atau falisitas pada browser tersebut, atau yang paling mudah adalah dengan check pada url berikut ini.

<http://viewportsizes.com/mine/>

<http://viewportsizes.com/>

Media query memiliki beberapa bagian, seperti Media Type, Logical Operator, dan Media Features, kurang lebih seperti di bawah ini.

**Berikut ini adalah contoh-contoh Media Type:**

|  |  |
| --- | --- |
| All | Used for all media type devices |
| Print | Used for printers |
| Screen | Used for computer screens, tablets, smart-phones etc. |
| Speech | Used for screenreaders that "reads" the page out loud |

**Berikut ini contoh beberapa logical operator:**

|  |  |
| --- | --- |
| And | dimana lebih dari satu statement tertulis dan seluruhnya harus dipenuhi agar selector di dalamnya bekerja. |
| ,(comma) | comma sifatnya seperti or di dalam programming language, salah satu content terpenuhi maka selector akan bekerja. |
| Not | Hasilnya akan sebaliknya, atau di negate |
| Only | mencegah browser lama yang belum memiliki fungsi media queries menggunakan seluruh selector yang ada pada media query statement. Ini bagus digunakan untuk lebih amannya, tetapi tidak akan ada pengaruh pada browser-browser modern. |

**Kalau yang ini adalah contoh-contoh dari Media features:**

|  |  |
| --- | --- |
| max-device-aspect-ratio | The maximum ratio between the width and the height of the device |
| max-device-height | The maximum height of the device, such as a computer screen |
| max-device-width | The maximum width of the device, such as a computer screen |
| max-height | The maximum height of the display area, such as a browser window |
| max-monochrome | The maximum number of bits per "color" on a monochrome (greyscale) device |
| max-resolution | The maximum resolution of the device, using dpi or dpcm |
| max-width | The maximum width of the display area, such as a browser window |
| min-aspect-ratio | The minimum ratio between the width and the height of the display area |
| min-color | The minimum number of bits per color component for the output device |
| min-color-index | The minimum number of colors the device can display |
| min-device-aspect-ratio | The minimum ratio between the width and the height of the device |
| min-device-width | The minimum width of the device, such as a computer screen |
| min-device-height | The minimum height of the device, such as a computer screen |
| min-height | The minimum height of the display area, such as a browser window |
| min-monochrome | The minimum number of bits per "color" on a monochrome (greyscale) device |
| min-resolution | For retina display only, minimum dpi (dot per inch) |
| min-width | The minimum width of the display area, such as a browser window |
| monochrome | The number of bits per "color" on a monochrome (greyscale) device |
| orientation | The orientation of the viewport (landscape or portrait mode) |
| overflow-block | How does the output device handle content that overflows the viewport along the block axis (added in Media Queries Level 4) |
| overflow-inline | Can content that overflows the viewport along the inline axis be scrolled (added in Media Queries Level 4) |
| resolution | The resolution of the output device, using dpi or dpcm |
| scan | The scanning process of the output device |
| update-frequency | How quickly can the output device modify the appearance of the content (added in Media Queries Level 4) |
| width | The viewport width |

Contoh pemakaian media-query kurang lebih seperti di bawah ini:

|  |
| --- |
| p{font-size:16px;}  @media only screen and (max-width: 780px) {  p{font-size:12px} } |

Code di atas artinya adalah bahwa dengan browser yang mendukung media query, bahwa device dengan layar, dimana layar tersebut memiliki lebar 780px ke bawah, maka semua ukuran font pada paragraph adalah 12 px, tapi apabila kondisi di luar itu, besarnya adalah 16px.

Apabila:

|  |
| --- |
| @media only screen and (max-width: 780px) and (min-width:320px) {  p{font-size:10px} } |

Artinya hanya browser dengan lebar antara 780px dan 320px yang memiliki font-size 10px.

# 3. Mobile Approach Strategy

Ini merupakan strategy mudah untuk para pemula membuat web dengan tampilan responsive, yaitu menggunakan strategy mobile approach. Mobile approach adalah strategy perencanaan dimana web designer dan front-end developer lebih mengutamakan pembuatan tampilan pada layar PC atau layar terbesar, baru memikirkan bagaimana membuat dan meng-code tampilan mobilenya.

Misalnya study casenya seperti ini, anda adalah seorang front-end developer yang diminta untuk bekerja sama dengan seorang web designer pada suatu project, lalu project tersebut meminta web aplikasi nya harus memiliki feature responsive, dimana web tersebut akan tampil bagus dan jelas pada pc dengan monitor 21 inch sampai dengan rata-mobile dengan ukuran 5 inch.

Waktu project sangat terbatas, dan front-end developer dengan designer diminta untuk bekerja separallel mungkin. Sedangkan waktu designer sangat terbatas untuk mengerjakan sebuah design, belum lagi designer harus menguji dan mempresentasikan karyanya, apakah disukai clients dan user atau tidak.

Maka strategy yang tepat adalah baik front-end developer dan designer mengerjakan tampilan dengan view port terbesarnya terlebih dulu. Di sini requirement meminta 21 inch – mobile 5 inch. Berarti kurang lebih dari lebar 1920px sampai 360px.

Contohnya seperti di bawah ini:

|  |
| --- |
| **.container{**  **Background-color:blue;**  **}**  @media only screen and (max-width:1024px){  .container{  Background-color: red;  }  }  @media only screen and (max-width:600px){  .container{  Background-color:yellow;  }  }  @media only screen and (max-width:360px){  .container{  Background-color:green;  }  } |

Pada awalnya front-end menulis css seperti biasa, misalnya ingin melakukan style sebuah tag dengan class container. Semua style ditulis sesuai dengan permintaan designer. Lalu ketika designer selesai mengerjakan design mobilenya, kita bisa membuat style sheet seperti di atas.

Daripada harus repot-repot menulis max-width dan min-width, kita bisa memanfaatkan max-width saja untuk mobile approach strategy. Sheet di atas bisa diartikan kalau secara default, background-color dari class container adalah blue. Tetapi ketika layar viewportnya mengecil sampai ukuran 1024px atau di bawahnya, maka background-color akan berwarna merah, apabila lebih kecil dari 600px, maka warna default (biru) dan yang memiliki viewport 1024px (warna merah) akan kalah dan di override dengan background-color:yellow.