Responsive Development

disruptive
digital
education

Samuel Traquina

Frontend Web Developer
Clever Advertising

Co-Founder + Sound Guy + Web Dev Dude + Co-host
mudopodcast.pt

Experiência profissional
2010-2015 | Fullstack Web Developer na Webcomum
2015-2016 | Fullstack Web Developer na Altronix
2016-2017 | Technology Visionary na Sabya
and also, freelance

Formação
Licenciatura Novas Tecnologias da Comunicação na UA

Website: http://trakina.net

Email: samuel@trakina.net

Skype: sam.traquina

Parte 2.

VIEWPORT, MEDIA QUERIES & SASS

Viewport

Viewport

- Necessário para que o layout seja responsive “across the board”

Viewport

- Necessário para que o layout seja responsive “across the board”
- Vamos culpar a Apple

Viewport

- Necessário para que o layout seja responsive “across the board”
- Vamos culpar a Apple
- Aparecimento de smartphones e, novamente, os 960

Viewport

- Necessário para que o layout seja responsive “across the board”
- Vamos culpar a Apple
- Aparecimento de smartphones e, novamente, os 960
- Prevenir temporariamente uma restrição que já não se aplica

Viewport

- Necessário para que o layout seja responsive “across the board”
- Vamos culpar a Apple
- Aparecimento de smartphones e, novamente, os 960
- Prevenir temporariamente uma restrição que já não se aplica
- Forma de dizer ao browser que medidas utilizar no render da página

Viewport

Tudo bem, mas o que é, mesmo?

Viewport

Tudo bem, mas o que é, mesmo?

É uma meta-tag que se coloca dentro da <head>, sendo a sua forma mais comum a seguinte:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Viewport

Os atributos da coisa

Viewport

Os atributos da coisa

width	Largura “virtual” do dispositivo
device-width	Largura real do ecrã do dispositivo
height	Altura “virtual” do dispositivo
device-height	Altura real do ecrã do dispositivo
initial-scale	Escala inicial em que a página é renderizada. 1.0 ou 1 mostram sem escalar, no tamanho disponível
minimum-scale	Escala mínima permitida ao utilizador. Usar 1.0 ou 1 para o utilizador não conseguir reduzir a escala
maximum-scale	Escala máxima permitida ao utilizador. Usar 1.0 ou 1 para o utilizador não conseguir aumentar a escala
user-scalable	Definir se o redimensionamento pode ser feito pelo utilizador. Os valores possíveis são “yes” e “no”.

Media Queries

Here comes the **big one**

Alguma vez usaram disto? HTML

```
<link rel="stylesheet" href="css/screen.css" media="screen">
<link rel="stylesheet" href="css/print.css" media="print">
```

Media Queries

Here comes the **big one**

Alguma vez usaram disto? HTML

```
<link rel="stylesheet" href="css/screen.css" media="screen">
<link rel="stylesheet" href="css/print.css" media="print">
```

As opções incluem:

aural, braille, handheld, print, projection, screen, tty, tv

Media Queries

Here comes the **big one**

Alguma vez usaram disto? HTML

```
<link rel="stylesheet" href="css/screen.css" media="screen">
<link rel="stylesheet" href="css/print.css" media="print">
```

As opções incluem:

aural, braille, handheld, print, projection, screen, tty, tv

Media Queries

Here comes the **big one**

Se quisermos complicar um bocado e adicionar mais verificações, para além de apenas o tipo de media

Media Queries

Here comes the **big one**

Podemos, depois, ir dar nisto: HTML

```
<link rel="stylesheet" href="css/small.css" media="screen
and (max-width: 700px)">
```

Media Queries

Here comes the **big one**

Pode, depois, ir dar nisto: HTML

```
<link rel="stylesheet" href="css/small.css" media="screen
and (max-width: 700px)">

<link rel="stylesheet" href="css/medium.css" media="screen
and (min-width: 701px) and (max-width: 1000px)">
```

Media Queries

Here comes the **big one**

Pode, depois, ir dar nisto: HTML

```
<link rel="stylesheet" href="css/small.css" media="screen
and (max-width: 700px)">

<link rel="stylesheet" href="css/medium.css" media="screen
and (min-width: 701px) and (max-width: 1000px)">

<link rel="stylesheet" href="css/large.css" media="screen
and (min-width: 1001px)">
```

Media Queries

Here comes the **big one**

Pode, depois, ir dar nisto: HTML

```
<link rel="stylesheet" href="css/small.css" media="screen
and (max-width: 700px)">

<link rel="stylesheet" href="css/medium.css" media="screen
and (min-width: 701px) and (max-width: 1000px)">

<link rel="stylesheet" href="css/large.css" media="screen
and (min-width: 1001px)">
```

E já temos aqui “mobile first”

Media Queries

Here comes the **big one**

Aqui podem usar-se, agregados com a expressão “and”:

- min-width, max-width, min-height, max-height, min-device-width, max-device-width
- device-aspect-ratio: 16/9;
- orientation: portrait;
- color
- min-color-index: 256;
- monochrome

Media Queries

Here comes the **big one**

Aqui podem usar-se, agregados com a expressão “and”:

- min-width, max-width, min-height, max-height, min-device-width, max-device-width
- device-aspect-ratio: 16/9;
- orientation: portrait;
- color
- min-color-index: 256;
- monochrome

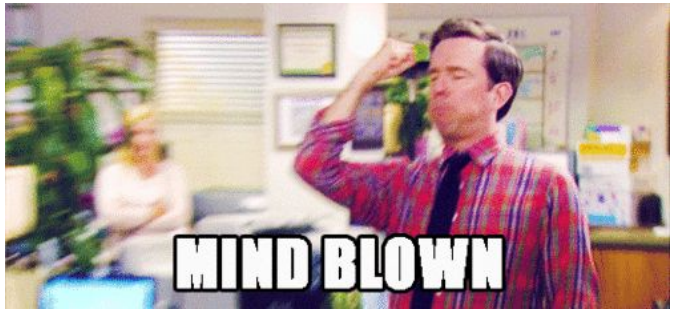
E podemos fazer tudo isto numa única folha de estilos

Media Queries

Here comes the **big one**

Aqui podem usar-se, agregados com a expressão “and”:

- min-width, max-width, min-height, max-height, min-device-width, max-device-width
- device-aspect-ratio: 16/9;
- orientation: portrait;
- color
- min-color-index: 256;
- monochrome



E podemos fazer tudo isto numa única folha de estilos

Media Queries

Here comes the **big one**

Como esta: CSS

```
@media screen and (max-width: 700px) {
     body {
          width: 100%;
               font-size: 16px;
     }
}
@media screen and (min-width: 701px) and (max-width: 1000px) {
     body {
          width: 90%;
               font-size: 14px;
     }
}
@media screen and (min-width: 1000px) {
     body {
          width: 80%;
               font-size: 14px;
     }
}
```

Media Queries

Here comes the **big one**

No exemplo seguinte:

- Definem-se os breakpoints
- Caso não se defina uma regra, assume-se a do breakpoint anterior
- Não definindo regras para outro media, retira-se o “screen”
- O primeiro breakpoint desaparece

Media Queries

Here comes the **big one**

Simplificando, com Progressive Enhancement: CSS

```
body {
     width: 100%;
          font-size: 16px;
}
@media (min-width: 701px) {
     body {
          width: 90%;
               font-size: 14px;
     }
}
@media (min-width: 1000px) {
     body {
          width: 80%;
     }
}
```

SASS

This is gonna hurt

SASS - Syntactically Awesome Style Sheets

- Pré-processadores (SASS, LESS, etc.)
- Gerar e organizar CSS
- Permitem utilizar lógica no CSS
- Vários ficheiros ".scss" geram um só ".css"
- Organização aumenta legibilidade

SASS

This is gonna hurt

Lógica?!

You're killing me, bro...

SASS

This is gonna hurt

- Variáveis ($a-minha-cor-primaria, $largura-minima)
- Estruturas de decisão (if, else)
- Ciclos de repetição (for, foreach, while)
- Mixins
- Hereditariedade
- Nesting
- Operadores
- Importações

SASS

This is gonna hurt

Variáveis em SASS para quê?

```
// imaginemos um gradiente

$type: 'radial'; // radial / linear
$from: #FFF; // cor de início do gradiente
$to: #000; // cor do fim do gradiente
```

SASS

This is gonna hurt

Estruturas de decisão

permitem-nos, de uma forma fácil, determinar o resultado de uma operação e executar algo em função desse mesmo resultado

SASS

This is gonna hurt

Estruturas de decisão em SASS para quê? IF

```
// imaginemos um gradiente

$type: 'radial'; // radial / linear
$from: #FFF; // cor de início do gradiente
$to: #000; // cor do fim do gradiente

// verificação de tipo de gradiente:
@if ($type == 'radial') {
          // aplica estilos de gradiente radial
}
```

SASS

This is gonna hurt

Estruturas de decisão em SASS para quê? IF ELSE

```
// imaginemos um gradiente

$type: 'radial'; // radial / linear
$from: #FFF; // cor de início do gradiente
$to: #000; // cor do fim do gradiente

// verificação de tipo de gradiente:
@if ($type == 'radial') {
          // aplica estilos de gradiente radial
} @else {
          // aplica estilos de gradiente linear
}
```

SASS

This is gonna hurt

Ciclos de repetição

permitem executar algo para cada iteração do ciclo

SASS

This is gonna hurt

Ciclos de repetição em SASS para quê? FOR

```
$gridcolumns: 12;
$gridgutter: 30px;
$gridwidth: 1200px;

@for $i from 1 through $gridcolumns {
          .col#{$i} {
               box-sizing: border-box;
               width: percentage($i/$gridcolumns);
               float: left;
               padding-left: $gridgutter / 2;
               padding-right: $gridgutter / 2;
          }
          .offset#{$i} {
               margin-left: percentage($i/$gridcolumns);
          }
}
```

SASS

This is gonna hurt

Ciclos de repetição em SASS para quê? WHILE

```
$gridcolumns: 12;
$gridgutter: 30px;
$gridwidth: 1200px;
@while $i <= $gridcolumns {
          .col#{$i} {
               box-sizing: border-box;
               width: percentage($i/$gridcolumns);
               float: left;
               padding-left: $gridgutter / 2;
               padding-right: $gridgutter / 2;
          }
          .offset#{$i} {
               margin-left: percentage($i/$gridcolumns);
          }
          $i: $i + 1;
}
```

SASS

This is gonna hurt

Mixins

Se temos algum pedaço de código que temos que executar repetidamente, utilizamos um mixin

Permite ainda, se o desejarmos, passar variáveis para utilizações diferentes do mesmo código

SASS

This is gonna hurt

Mixins em SASS para quê?

```
// queremos algo que nos calcule gradientes:
@mixin gradient($type,$from,$to){
    background: $from;
    @if ($type == 'radial') {
        background: radial-gradient($from, $to);
    } @else {
        background: linear-gradient($from, $to);
    }
}
// chamamos assim no elemento:
.box { @include gradient( 'linear', #FFF, #000 ); }
```

SASS

This is gonna hurt

Hereditariedade

Um elemento poder “extender” outro, ou seja, assumir os estilos de outro elemento, para além de manter os seus.

SASS

This is gonna hurt

Hereditariedade em SASS para quê?

```
.textopreto{
          color: black;
}
.textobold{
          font-weight: bold;
}

#omeutitulo{
          @extend .textopreto;
          @extend .textobold;
}
```

SASS

This is gonna hurt

Nesting

Pode seguir a mesma estrutura que o HTML: este elemento dentro daquele dentro de outro.

SASS

This is gonna hurt

Nesting em SASS para quê? **Se o HTML é assim:**

```
<nav>
        <ul>
            <li><a href="">el link</a></li>
            <li><a href="">el link</a></li>
        </ul>
</nav>
```

SASS

This is gonna hurt

Nesting em SASS para quê? No CSS temos isto:

```
nav {
          ul {
               margin: 0;
               padding: 0;
               list-style: none;
               li {
                    display: inline-block;
                    a {
                         display: block;
                         padding: 6px 12px;
                         text-decoration: none;
                    }
               }
          }
}
```

SASS

This is gonna hurt

Operadores

Podem usar-se as quatro operações básicas(+ / - *), mais a divisão de resto zero (%)

SASS

This is gonna hurt

Operadores em SASS para quê?

```
article {
  float: left;
  width: 600px / 960px * 100%;
}
```

SASS

This is gonna hurt

Importação

Agregar outros ficheiros a um ficheiro inical, para ser gerado apenas um único CSS

SASS

This is gonna hurt

Importação em SASS para quê?

```
// ficam na mesma pasta com um "_" antes do nome "chamado"
@import 'reset'; // "_reset.scss"
@import 'vars'; // "_vars.scss"
@import 'mixins'; // "_mixins.scss"
@import 'grid'; // "_grid.scss"
@import 'fa_font-awesome'; // "_fa_font-awesome.scss"
@import 'text'; // "_text.scss"
@import 'containers'; // "_containers.scss"
@import 'navbar'; // "_navbar.scss"
```

Posto isto, bater código!

AGAAAAIN?!

Agora a versão mobile a partir de uma desktop fornecida.

Sem ajudas de tamanhos e cores, tudo é para ver a partir do Invision.