Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento de Sites para Web (DSW)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

Responsive Web Design

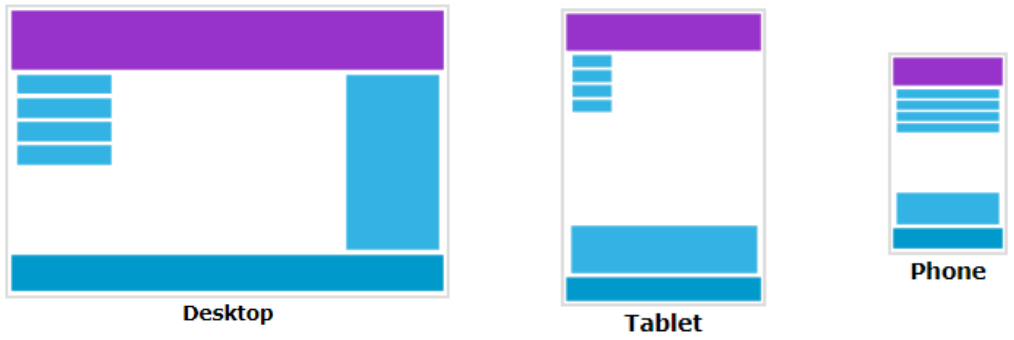
INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

Responsive Web Design

O RWD visa fazer com que as páginas Web tenham uma **boa aparência** nos **diferentes dispositivos** que as acessam.

Design Responsivo é uma técnica de estruturação HTML e CSS, ***que consiste em adaptar o site ao browser do usuário sem que seja necessário definir várias folhas de estilos específicas para cada resolução****, ou seja, é um tipo de design onde o layout fica fluído e variante de acordo com a resolução do usuário.*

- **RWD consiste na utilização de CSS + HTML para redimensionar, ocultar, reduzir, ampliar ou mover o conteúdo das páginas Web.**
- **O objetivo é deixar a disposição do conteúdo bem em qualquer tela.**

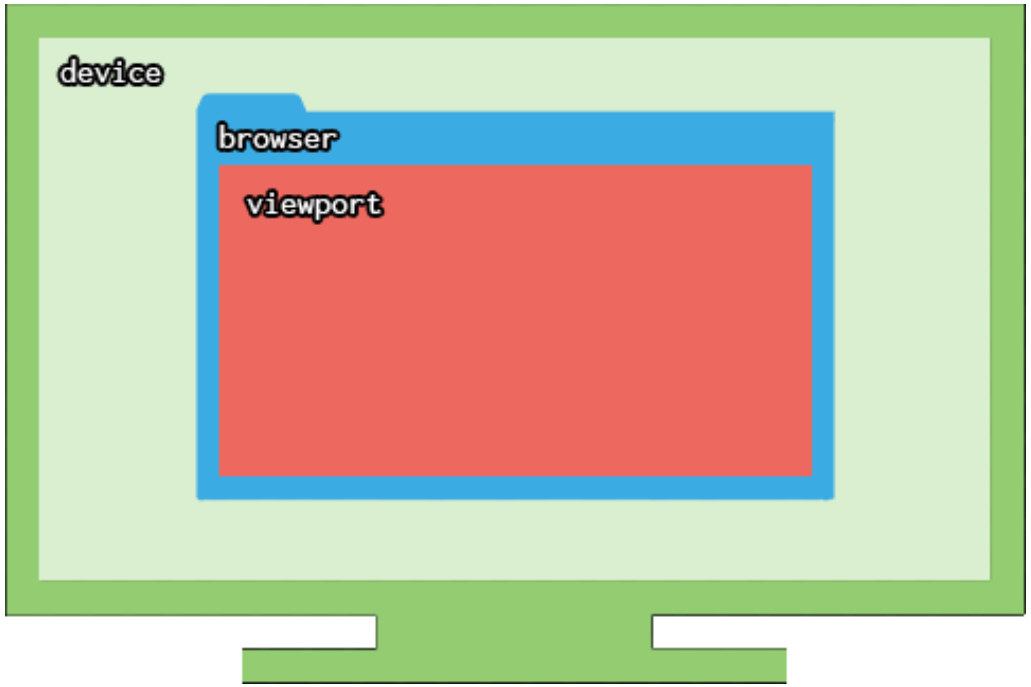


Objetivo: criar do zero, um layout responsivo semelhante a este.

Exemplo.html

Janela de exibição - `viewport`

É a **área visível pelo usuário** em uma página web.



A `viewport` **varia de dispositivo para dispositivo** (computadores, tablets, *smartphones*...)

Ela também varia quando o tamanho navegador **é redimensionado**.

História

Antes dos tablets e smartphones, as páginas da web **eram projetadas apenas para telas de computador**, e era comum que as páginas tivessem um ***design* estático e um tamanho fixo**.

Desktop/Laptop

Quando a navegação na Internet por meio de dispositivos móveis começou, **as páginas web de tamanho fixo eram muito grandes** para caber nas janelas de exibição (`viewports`).

Para corrigir isso, os navegadores nesses dispositivos **reduziram as páginas web para caber nas telinhas** (miniaturização dos sites)

Mobile

Bad UX

tag <meta>

O HTML5 introduziu um método para permitir ao desenvolvedor **controlar a área visível do navegador (viewport)**, por meio da tag **<meta>**.

A tag <meta> define metadados sobre um documento HTML.

Ao incluir o elemento <meta> abaixo, fornecemos ao navegador instruções sobre como controlar as dimensões e a escala da página.

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

Configura que a largura da página deve seguir a largura da tela do dispositivo

Define o nível de zoom inicial quando a página é carregada pela primeira vez no navegador

Com `viewport` configurada

Sem `viewport` configurada

Se **não** existe a **`viewport`** declarada, o **dispositivo móvel irá encolher o site** para mostrá-lo com a largura de 980px.

Sites não responsivos ficam como miniaturas quando acessados por um dispositivo móvel.

Em um desktop ou notebook, o padrão é que o **`viewport` seja a largura em pixels do seu navegador**. Ao redimensionar o browser, verá menos do site. Logo, o **`viewport`** será menor.

E nos dispositivos móveis?

O funcionamento é diferente.

Basicamente, não existe como **redimensionar a tela do navegador do celular**.
Assim, os aplicativos ocupam 100% da largura.

- **No começo da era dos smartphones, quando não se falava em design responsivo, foi preciso contornar este problema.**

- Era trabalhoso abrir um site e ver apenas uma porção pequena dele. O site teria **diversas barras de rolagem e o conteúdo apresentado seria muito pequeno**.

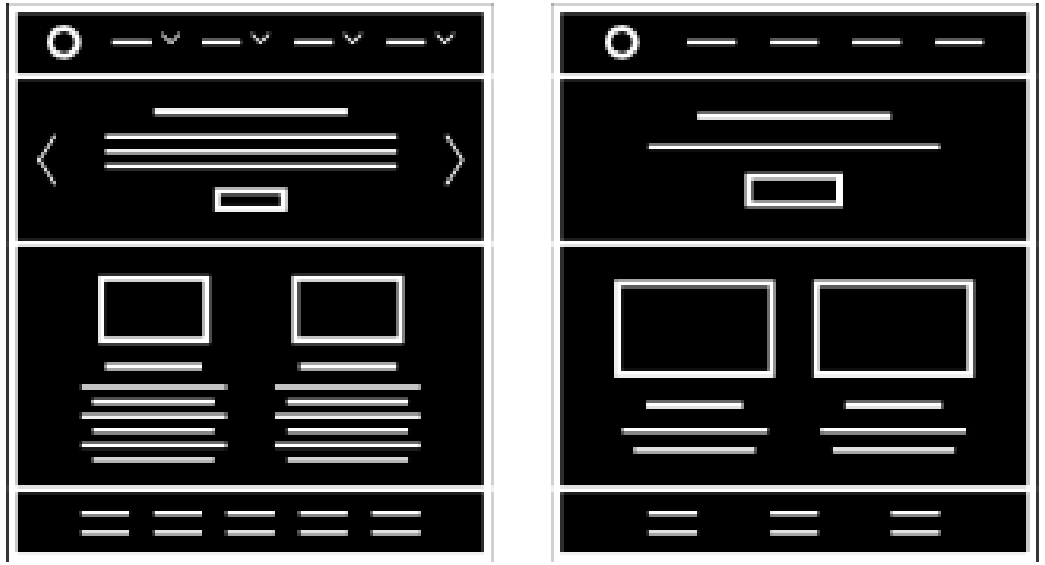
Pensando nisso, foi padronizado que o **`viewport` de dispositivos móveis seria de 980px**.

Assim, se não existe a meta-tag **`viewport`** declarada, o **dispositivo móvel encolhe o site para mostra-lo com a largura de 980px**.

`viewport` – Boas práticas

Dimensionar o conteúdo para a `viewport`

Usuários estão acostumados a rolar os sites verticalmente , **mas não horizontalmente**!



Forçar os usuários a rolar o conteúdo horizontalmente ou diminuir o zoom para ver toda a página da web, resultará em uma **UX ruim**.

Outras práticas

NÃO utilizar elementos grandes de largura fixa

Exemplo: se uma imagem for exibida com uma largura maior do que a `viewport`, isso pode fazer com que o `viewport` role horizontalmente. **Ajuste o conteúdo para caber na largura da `viewport`.**

NÃO deixar o conteúdo depender de uma largura de `viewport` específica para renderizar bem

Como as dimensões da tela e a largura em pixels CSS variam amplamente entre os dispositivos, **o conteúdo não deve depender de uma largura de uma `viewport` específica para renderizar bem.**

Utilizar *Media Queries* para aplicar estilos diferentes para telas pequenas e grandes

A definição de larguras absolutas grandes para os elementos da página fará com que o elemento seja muito largo para a `viewport` em um dispositivo menor. **Considere o uso de valores de largura relativos, como width: 100%.**

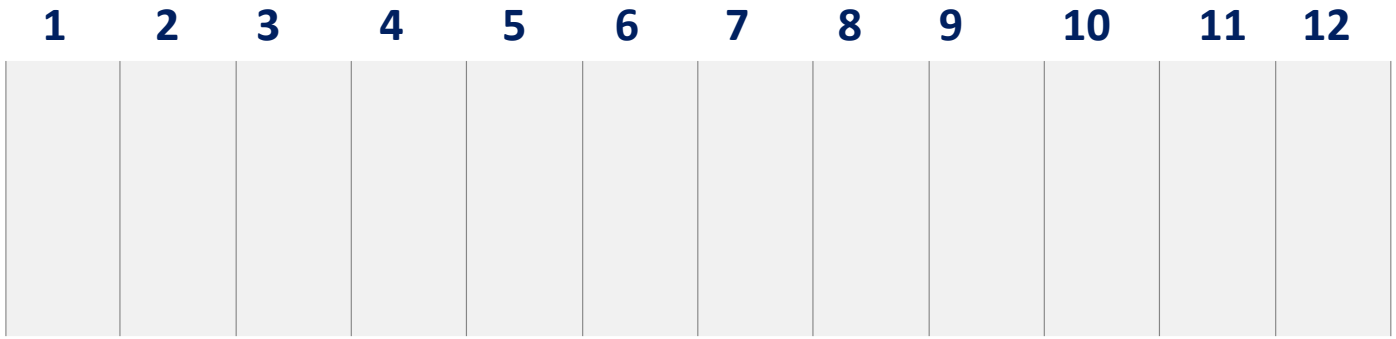
Cuidado ao usar grandes valores de posicionamento absolutos. Isso pode fazer com que o elemento fique fora da `viewport` em dispositivos pequenos.

CSS

grid-view

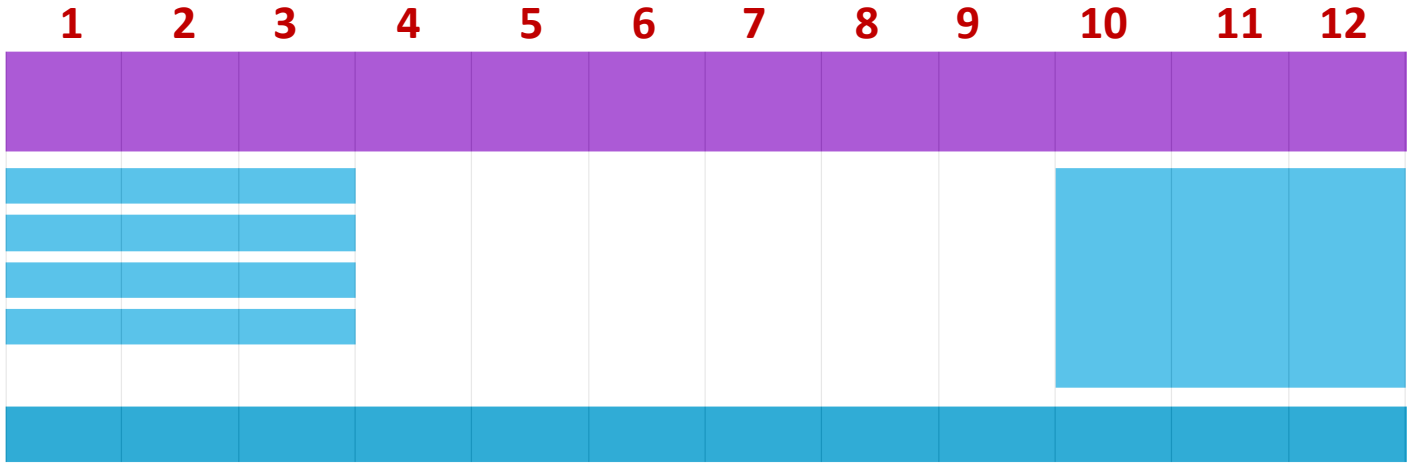
O que é uma `grid-view`

Muitas páginas web são baseadas em grade. Ou seja, a **página é dividida em colunas**.



Uma `grid-view` responsiva geralmente tem 12 colunas e uma largura total de 100%, e irá encolher e expandir conforme a janela do navegador é redimensionada.

`Grid-view` é útil ao projetar páginas da web, pois simplifica a **colocação de elementos na página**.



Construindo uma grid-view responsiva

Primeiramente é preciso garantir que as propriedade **padding** e **border** sejam **incluídos** na largura e altura total dos elementos.

Por padrão, altura e largura dos elementos são calculadas dessa forma:

```
largura = width  + padding + border
altura  = height + padding + border
```

Isso significa que ao configurar `width/height` de um elemento, **ele frequentemente parece maior do que foi configurado**, já que `border` e `padding` são adicionados a altura/largura do elemento.

Exemplo:

- Tendo **`.box {width: 350px}`**
- Ao aplicar uma propriedade **`{border: 10px solid black;}`**
- O resultado no navegador será **`.box {width: 370px;}`**

Para alterar isso, Podemos usar a propriedade **`box-sizing`**, a qual permite incluir o **`padding`** e a **`border`** na largura e altura total de um elemento

```
* {
    box-sizing: border-box;
}
```

O seletor universal **`*`** aplica a regra CSS a todos os elementos HTML.

`box-sizing` indica como o navegador calcula as dimensões dos elementos

`border-box` é usada para incluir o **`padding`** e o **`border`** na largura total do elemento.


```
<body>
      <div class="header">
        <h1>Pelotas</h1>
      </div>

      <div class="menu">
        <ul>
            <li>Como chegar?</li>
            <li>Cidade</li>
            <li>Lagoa dos Patos</li>
            <li>FENADOCE</li>
        </ul>
      </div>

      <div class="main">
        <h1>Pelotas</h1>
        <p>Município da zona sul do RS, Brasil. </p>
        <p>Redimensione a janela do navegador.</p>
      </div>
</body>
```

a) Faz com os elementos HTML tenham a propriedade `box-sizing` definida como `border-box`

```
* {
    box-sizing: border-box;
}
```

b) Página com duas colunas. A primeira ocupa 25% e a segunda 75% do espaço.

```
.header {
  border: 1px solid red;
  padding: 15px;
}

.menu {
  width: 25%;
  float: left;
  padding: 15px;
  border: 1px solid blue;
}

.main {
  width: 75%;
  float: left;
  padding: 15px;
  border: 1px solid darkgrey;
}
```

c) `float` tira os elementos do eixo normal e faz com que as colunas se alinhem lado a lado..

Este exemplo é adequado se a página tiver apenas 2 colunas.

Mas, a ideia é usar uma **grid-view** responsiva com 12 colunas,
para ter mais controle sobre a página web.

div header

div menu

div main

Exemplo_02.html

No CSS

a) Calcular a porcentagem para uma coluna:
E atribuir esses tamanhos a cada uma delas.
100% / 12 colunas = 8,33%

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
...
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

b) Criar uma classe para cada uma das 12 colunas, **class="col-"** e um número que define quantas colunas a seção deve abranger.

[] seletor de atributo:
Seleciona elementos que tenham pelo menos uma ocorrência de `col-` como seu valor de classe.

```
[class*="col-"] {
  float: left;
  padding: 15px;
  border: 1px solid red;
}
```

c) Criar e aplicar um estilo em que essas colunas devem **flutuar para a esquerda**:

No HTML

```
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-9">...</div> <!-- 75% -->
</div>
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-6">...</div> <!-- 50% -->
  <div class="col-1">...</div> <!- 8.33% -->
  <div class="col-2">...</div> <!- 16.66% -->
</div>
```

a) Cada linha deve ser envolvida em um `<div>`.

b) O número de colunas dentro de uma linha deve sempre somar 12.

No CSS

```
[class*="col-"] {
  float: left;
  padding: 15px;
  border: 1px solid red;
}
```

c) Aplicar o estilo aos elementos selecionados.

d) As colunas dentro de uma linha flutuam para a esquerda, sendo retiradas do fluxo normal da página.

e) Dessa forma os outros elementos serão colocados como se as colunas não existissem.

f) Para evitar isso, podemos adicionar um estilo (`ClearFix`) que limpa o fluxo.

```
.row::after {
  content: "";
  clear: both;
  display: table;
}
```

Estilos adicionais para melhorar aparência


```
html {
  font-family: "Lucida Sans", sans-serif;
}

.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

Exemplo_03.html

```
.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color :#33b5e5;
  color: #ffffff;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
  background-color: #0099cc;
}
```

CSS

Media Queries

CSS2 *Media Types*

CSS2 introduziu o conceito de Tipos de Mídia (***Media Types***).

A regra `@media` possibilitou definir diferentes **regras de estilo para diferentes tipos de mídia**.

Tornou possível termos **conjuntos de regras de estilo** distintos para **telas de computador**, **impressoras**, **dispositivos portáteis**, **televisão** e outros.

Infelizmente, esses tipos de mídia nunca tiveram muito suporte, além do tipo de mídia: **impressão**.

CSS3 *Media Queries*

CSS3 introduziu o conceito de ***Media Queries***.

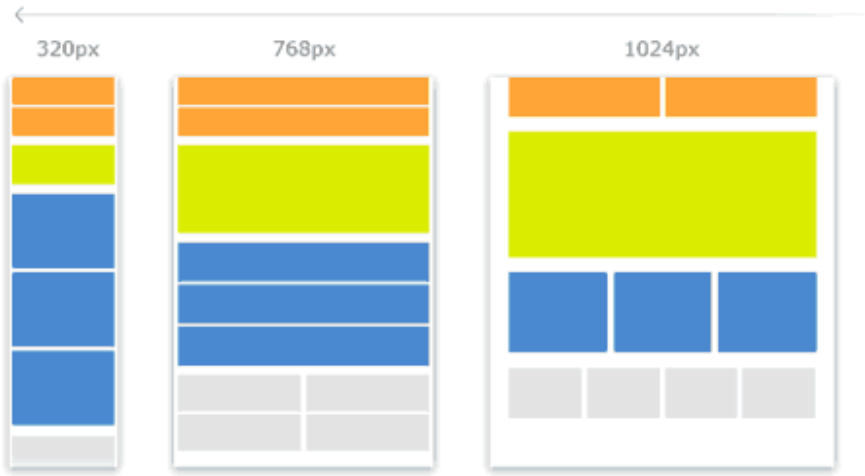
As **Media Queries** do CSS3 estenderam a ideia de **Media Types** do CSS2

Ao invés de procurar um **tipo de dispositivo**, as MQ focam na **capacidade do dispositivo**.

MQ é uma técnica para fornecer folhas de estilo personalizadas para desktops, laptops, tablets e smartphones.

Estas consultas (MQ) podem verificar coisas, como:

1. **Largura e altura da janela de visualização;**
2. **Largura e altura do dispositivo;**
3. **Orientação o dispositivo está (paisagem/retrato);**
4. **Resolução;**



Sintaxe

Uma MQ consiste em um **tipo de mídia (*media type*)** e pode conter **uma ou mais expressões**, que resultarão em **Verdadeiro** ou **Falso**.

```
@media
not|only mediatype and (expressões) {
  Código CSS;
}
```

O resultado da consulta é **verdadeiro se o tipo de mídia especificado corresponder ao tipo de dispositivo** no qual o documento está sendo exibido **e todas as expressões na consulta de mídia forem verdadeiras**.

Quando uma **MQ é verdadeira**, a folha de estilo ou as regras de estilo correspondentes são aplicadas, seguindo as regras normais em cascata.

CSS3 *Media Types*

Valor	Descrição
`all`	Usado para todos *media type*
`print`	Usado para impressoras
`screen`	Usado para telas de computadores, tablets, smartphones, etc.
`speech`	Usado para *screenreaders* que “lêem" a página

Exemplo de *Media Query*

Uma maneira de usar consultas de mídia é ter uma **seção CSS alternativa** dentro de sua folha de estilo.

Exemplo

```
<body>
    <h1>Redimensione a janela para observar!</h1>
    <p>
       A MQ somente será aplicada se o media type
       for screen e a janela for menor que 680px.
    </p>
</body>
```

```
body {
    background-color: yellow;
  }

@media screen and (min-width: 680px) {
    body {
      background-color: lightblue;
    }
}
```

Altera a cor de fundo se a janela de visualização tiver 680 pixels de largura ou mais.

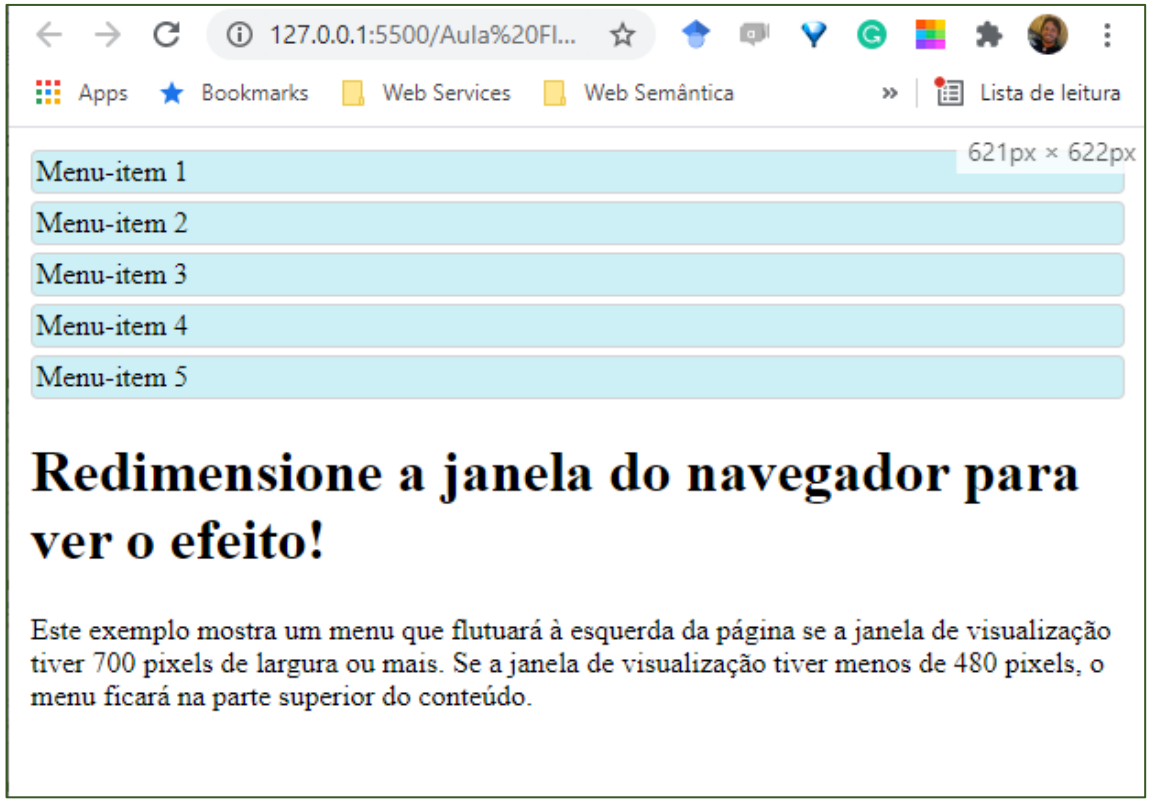
```html
<body>
    <div class="wrapper">
        <div id="leftsidebar">
          <ul id="menulist">
            <li class="menuitem">Menu-item 1</li>
            <li class="menuitem">Menu-item 2</li>
            <li class="menuitem">Menu-item 3</li>
            <li class="menuitem">Menu-item 4</li>
            <li class="menuitem">Menu-item 5</li>
          </ul>
        </div>

        <div id="main">
          <h1>Redimensione o browser!</h1>
          <p>
             Exemplo..... Textão.
          </p>
        </div>
      </div>
</body>
```

O menu que flutuará à esquerda **se a janela de visualização tiver 700 pixels de largura ou mais**. Se a janela de visualização tiver menos de 480 pixels, o menu ficará na parte superior do conteúdo.

```css
.wrapper {overflow: auto;}

#main {margin-left: 4px;}

#leftsidebar {
  float: none;
  width: auto;
}

#menulist {
  margin: 0;
  padding: 0;
}

.menuitem {
  background: #CDF0F6;
  border: 1px solid #d4d4d4;
  border-radius: 4px;
  list-style-type: none;
  margin: 4px;
  padding: 2px;
}

@media screen and (min-width: 700px) {
  #leftsidebar {width: 200px; float: left;}
  #main {margin-left: 216px;}
}
```

Exemplo

Efeitos da MQ no exemplo 2

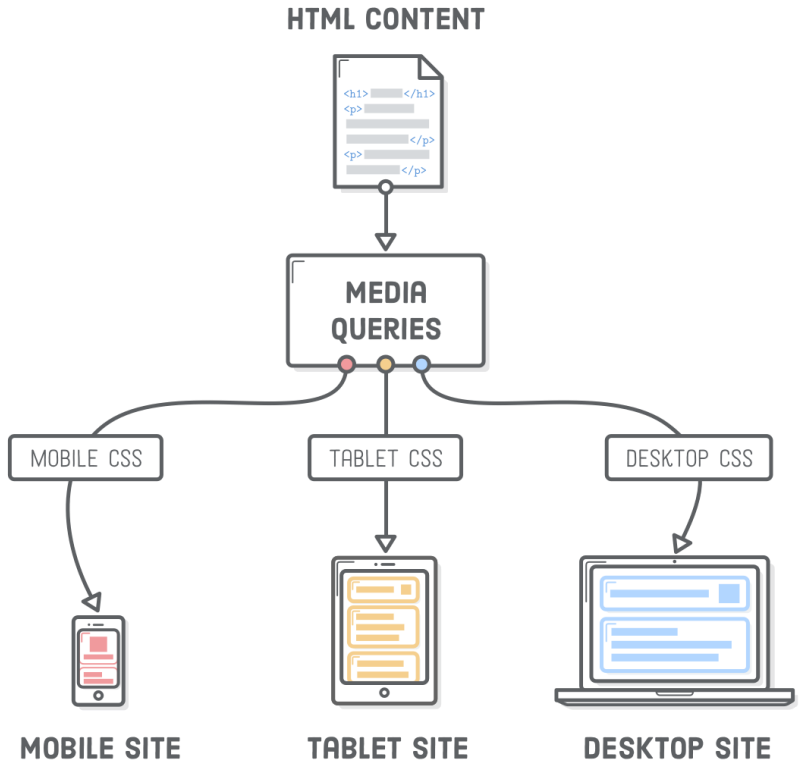


Media Queries – recordando...

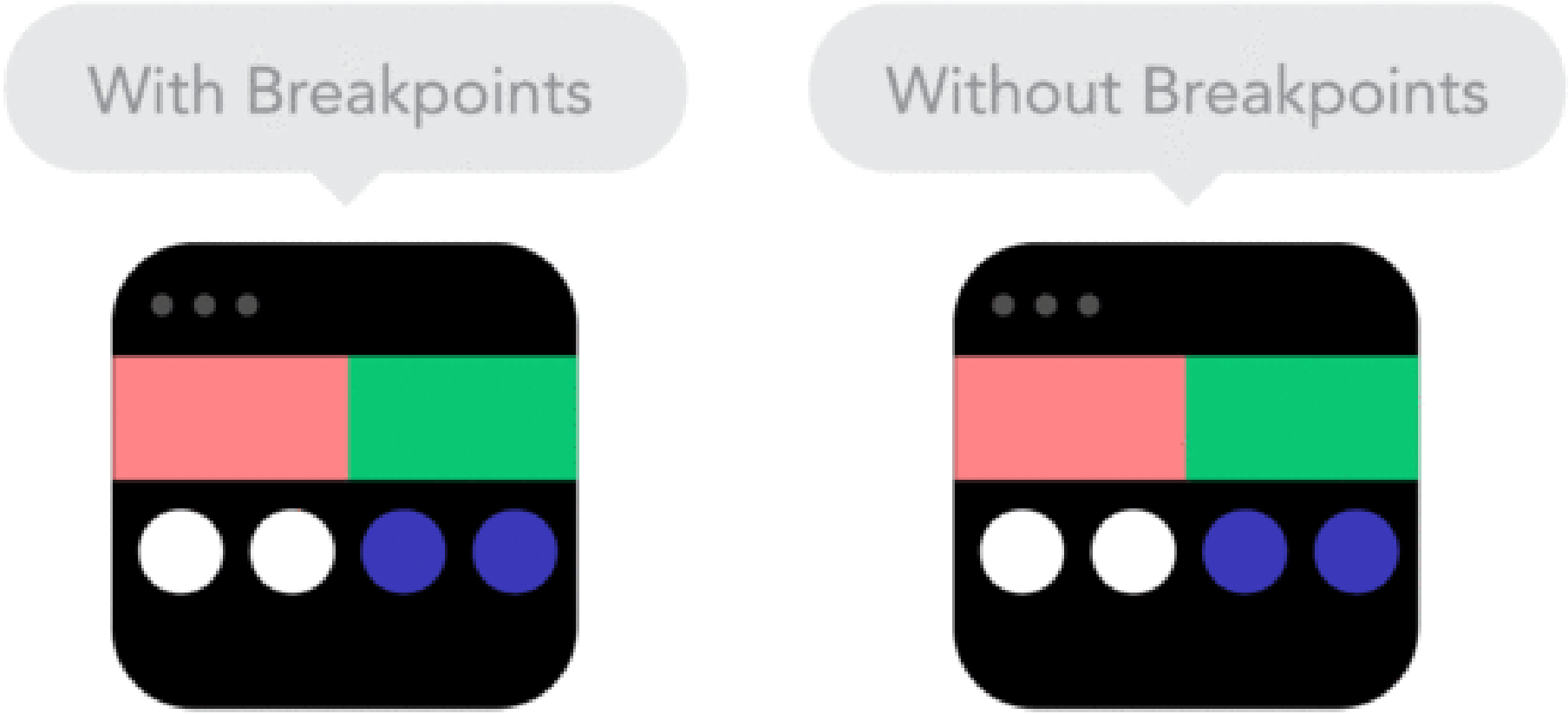
Usa a regra `@media` para incluir um bloco de propriedades CSS apenas se uma determinada **condição for verdadeira**.

```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

*** Se a janela do navegador tiver 600 px ou menos, a cor de fundo será azul claro**



Media Queries Breakpoints



Media Queries breakpoints

```
/* Para desktops: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Desktop

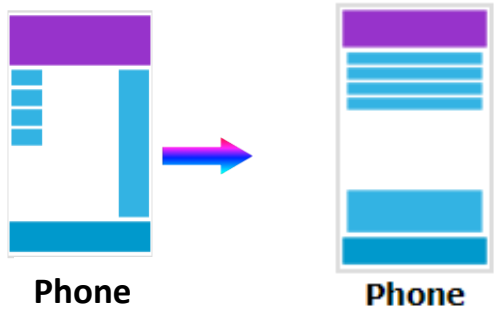
Phone

Breakpoint

Ponto a partir da onde, partes específicas do site se comportarão de maneira diferente em cada lado do *breakpoint*.

Media Queries breakpoints

```
/* Para desktops: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```




```
@media only screen and (max-width: 768px) {
  /* Para smartphones: */
  [class*="col-"] {
    width: 100%;
  }
}
```

Exemplo.html

MQ
Quando a janela do navegador for menor que 768px, cada coluna deve ter uma largura de 100%:

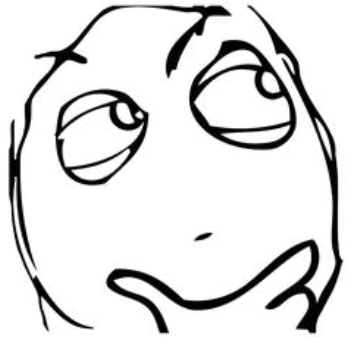
Mobile First

Projetar para dispositivos móveis antes de projetar para desktops ou qualquer outro dispositivo

Isso torna a **exibição da página mais rápida em dispositivos menores**.

Quais modificações precisamos fazer?

```
@media only screen and (max-width: 768px) {
  /* Para smartphones: */
  [class*="col-"] {
    width: 100%;
  }
}
```



```
/* Para desktops: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

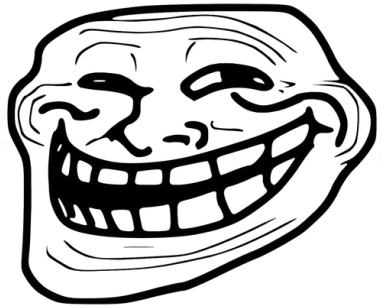
Mobile First

Projetar para dispositivos móveis antes de projetar para desktops ou qualquer outro dispositivo

Isso torna a **exibição da página mais rápida em dispositivos menores**.

Quais modificações precisamos fazer?

```
@media only screen and (max-width: 768px) {
  /* Para smartphones: */
  [class*="col-"] {
    width: 100%;
  }
}
```



```
@media only screen and (min-width: 768px) {
  /* Para desktops: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

E os tablets?

a) Adicionar mais uma `Media Query` com *breakpoint* de 600px.

b) Adicionar um conjunto de novas classes para dispositivos maiores que 600px (mas menores que 768px):

```
@media only screen and (min-width: 600px) {
   /* Para tablets: */
   .col-s-1 {width: 8.33%;}
   .col-s-2 {width: 16.66%;}
   .col-s-3 {width: 25%;}
   .col-s-4 {width: 33.33%;}
   .col-s-5 {width: 41.66%;}
   .col-s-6 {width: 50%;}
   .col-s-7 {width: 58.33%;}
   .col-s-8 {width: 66.66%;}
   .col-s-9 {width: 75%;}
   .col-s-10 {width: 83.33%;}
   .col-s-11 {width: 91.66%;}
   .col-s-12 {width: 100%;}

}
```


```
<div class="row">
  <div class="col-3 col-s-3">...</div>
  <div class="col-6 col-s-9">...</div>
  <div class="col-3 col-s-12">...</div>
</div>
```

Parece estranho ter dois conjuntos de classes idênticas, mas isso permite decidir o que acontecerá com as colunas em cada *breakpoint*

Exemplo.html

```
/* Para smartphones: */
[class*="col-"]{
  width: 100%;
}
```

```
@media only screen and (min-width: 600px){
  /* Para tablets: */
  .col-s-1 {width: 8.33%;}
  .col-s-2 {width: 16.66%;}
  .col-s-3 {width: 25%;}
  .col-s-4 {width: 33.33%;}
  .col-s-5 {width: 41.66%;}
  .col-s-6 {width: 50%;}
  .col-s-7 {width: 58.33%;}
  .col-s-8 {width: 66.66%;}
  .col-s-9 {width: 75%;}
  .col-s-10 {width: 83.33%;}
  .col-s-11 {width: 91.66%;}
  .col-s-12 {width: 100%;}
}
```

```
@media only screen and (min-width: 768px){
  /* Para desktops: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

Breakpoints típicos

Existem **muitas telas e dispositivos com diferentes alturas e larguras**, por isso é difícil criar um ponto de interrupção exato para cada dispositivo.

Para simplificar, é possível segmentar cinco grupos principais:

```
/* Dispositivo extra pequenos (fones, 600px ou menos) */
@media only screen and (max-width: 600px) {...}

/* Dispositivo pequenos (tablets retrato e fones grandes, 600px ou mais) */
@media only screen and (min-width: 600px) {...}

/* Dispositivos médios (tablets paisagem, 768px ou mais) */
@media only screen and (min-width: 768px) {...}

/* Dispositivos grandes (laptops/desktops, 992px ou mais) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (laptops e desktops grandes, 1200px e mais) */
@media only screen and (min-width: 1200px) {...}
```

Orientação: Retrato ou Paisagem

Media Queries também podem ser usadas para alterar o layout de uma página, dependendo da **orientação do navegador**.

A página da web terá um fundo azul claro se a orientação estiver no modo paisagem:

```
@media only screen and (orientation: landscape) {
  body {
    background-color: lightblue;
  }
}
```

Esconder elementos, em diferentes tamanhos de telas.

```
/* Se a tela é 600px ou menos, esconde o elemento*/
@media only screen and (max-width: 600px) {
  div.example {
    display: none;
  }
}
```

Alterar tamanho de **fonte**.

```
/* Se a tela é 601px ou maior, muda font-size da <div> to 80px */
@media only screen and (min-width: 601px) {
  div.example {
    font-size: 80px;
  }
}

/* Se a tela é 600px ou menos, muda font-size da <div> para 30px */
@media only screen and (max-width: 600px) {
  div.example {
    font-size: 30px;
  }
}
```

CSS

Imagens

Imagens responsivas

Definir a propriedade da **width** com uma **porcentagem** e a propriedade da **height** como "**auto**", torna a **imagem responsiva**.

```
<img src="mercado.jpg" width="460" height="345">
```

```
img {
    width: 100%;
    height: auto;
}
```

Problema: imagem pode ser ampliada para ficar maior que seu tamanho original.

Exemplo.html

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

Solução: Usar a propriedade **max-width** em 100% Imagem será reduzida, mas nunca será maior do que seu tamanho original

Exemplo.html

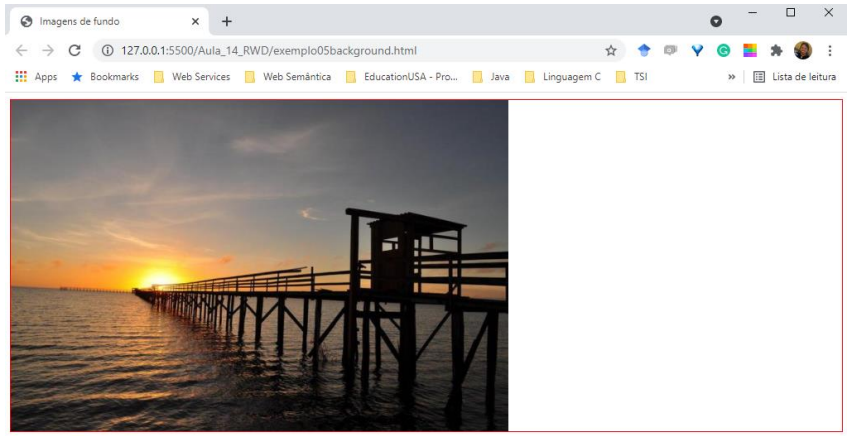
Redimensionamento de imagens de fundo

***Backgrounds* também podem responder ao redimensionamento e dimensionamento.**

```
div {
      width: 100%;
      height: 400px;
      background-image: url('laranjal.jpg');
      background-repeat: no-repeat;
      background-size: contain;
      border: 1px solid red;
}
```

1) Propriedade **"background-size: contain"** *background* será dimensionado e tentará se ajustar à área de conteúdo.

A imagem manterá sua proporção entre largura e altura.



Exemplo.html

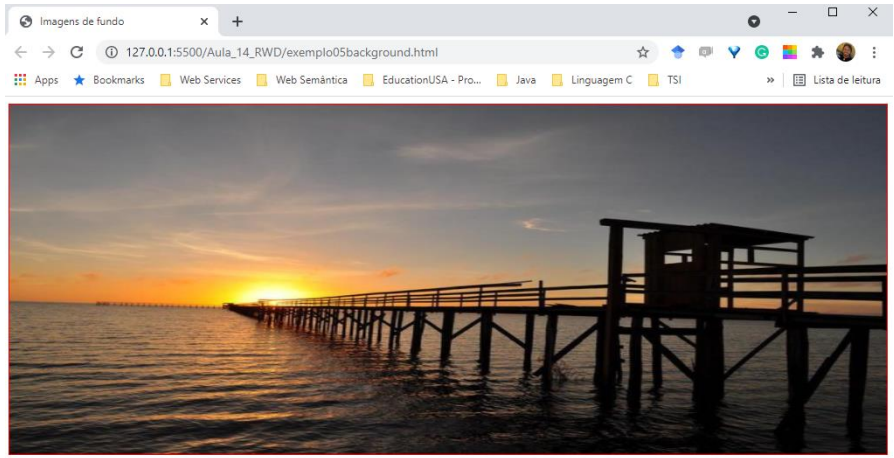
Redimensionamento de imagens de fundo

```
div {
      width: 100%;
      height: 400px;
      background-image: url('laranjal.jpg');
      background-repeat: no-repeat;
      background-size: 100% 100%;
      border: 1px solid red;
}
```

2) A propriedade **background-size: 100% 100%**

a imagem de plano de fundo será esticada para cobrir toda a área de conteúdo

Exemplo.html

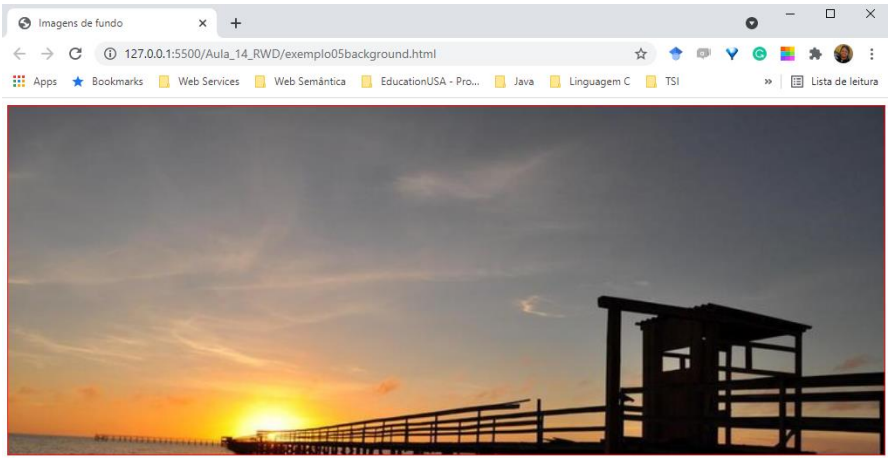


Imagens de fundo

```
div {
      width: 100%;
      height: 400px;
      background-image: url('laranjal.jpg');
      background-repeat: no-repeat;
      background-size: cover;
      border: 1px solid red;
}
```

A propriedade **background-size: cover** a imagem de fundo será dimensionada para cobrir toda área do conteúdo.

cover mantém a proporção, mas parte da imagem pode ser cortada.



Exemplo.html

Imagens diferentes para dispositivos diferentes

Uma imagem grande pode ser perfeita em uma tela grande de computador, **mas inútil em um dispositivo menor**.

```
/* Largura menor que 550px */
body {
  background-repeat: no-repeat;
  background-image: url('laranjal_pequeno.jpg');
}

/* Largura de 550px e maiores */
@media only screen and (min-width: 550px) {
  body {
     background-image: url('laranjal.jpg');
  }
}
```

Exemplo_06.html

É possível usar `media queries` para verificar a largura do navegador, e assim utilizar imagens adequadas.

min-device-width


```
/* Largura menor que 550px */
body {
  background-repeat: no-repeat;
  background-image: url('laranjal_pequeno.jpg');
}

/* Largura de 550px e maiores */
@media only screen and (min-device-width: 550px) {
  body {
     background-image: url('laranjal.jpg');
  }
}
```

min-device-width

Verifica a **largura do dispositivo**, em vez da largura do navegador.

Com esse método, a imagem não muda ao redimensionar a janela do navegador.

Exemplo_07.html

CSS

Elemento `picture`

Elemento `<picture>`

Oferece mais **flexibilidade** na especificação de recursos de imagem.

Seu uso mais comum é para imagens em designs responsivos.

Em vez de ter uma imagem que é redimensionada com base na largura do **`viewport`**, diversas **imagens podem ser projetadas para preencher melhor a `viewport`**.

Funciona de forma semelhante aos elementos **`<video>`** e **`<audio>`**

Diferentes fontes são configuradas, e a primeira fonte que se encaixa nas preferências é usada.

Elemento <picture>

Atributo obrigatório e define a origem da imagem.

Atributo opcional, o qual aceita as MQ que podem ser encontradas na regra @media.

```
<picture>
    <source srcset="laranjal_pequeno.jpg" media="(max-width: 550px)">
    <source srcset="laranjal.jpg">
    <img src="laranjal.jpg" alt="Praia do Laranjal" style="width:auto;">
</picture>
```

É preciso definir um elemento <img> para browsers que não suportam <picture>

Exemplo08.html

CSS

Vídeos

<video>


```
<video width="700" controls>
    <source src="ovos.mp4" type="video/mp4">
    <source src="ovos.3gp" type="video/3gpp">
    Seu navegador não suporta elemento vídeo do HTML.
</video>
```

width for definida como 100%, o player de vídeo será responsivo e aumentará ou diminuirá

```
video {
  width: 100%;
  height: auto;
}
```

max-width em 100%, o player diminuirá, mas nunca aumentará para ser maior do que seu tamanho original

```
video {
  max-width: 100%;
  height: auto;
}
```

Exemplo_video.html

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento de Sites para Web (DSW)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

CSS
Responsive Web Design

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas