Bootstrapp

Norberto Moreira


www.getbootstrap.com
Bootstrap is Front-end Framework
HTML, CSS, and JS framework for
developing responsive, mobile first projects on the web.

O que é?

Bootstrap nada mais é que um framework Front End utilizados por milhares de desenvolvedores web pelo mundo. A partir dele, muitas etapas do desenvolvimentos web se tornam mais rápidas e dinâmicas, pois já trazem consigo diversos elementos prontos.

Ele é uma ferramenta gratuita para desenvolvimento **HTML**, **CSS** e JS. Portanto, é uma ferramenta de código fonte aberto para todos. Contudo, hoje em dia, é o framework Front End mais conhecido e mais utilizado pelo mundo.

Importância

Conforme explicado anteriormente, o Bootstrap é hoje um dos frameworks/Front End mais utilizados pelo mundo inteiro. Portanto, sua importância para a criação de websites é de enorme referência.

Seus padrões seguem os princípios de usabilidade e as tendências de design para interfaces. Além disso, sua padronização permite que os sites tenham um visual atraente, ou seja, uma forma de **criar páginas** esteticamente agradáveis.

Como surguiu?

Como uma tentativa de resolver uma incompatibilidade dentro da própria equipe, os engenheiros Jacob Thorton e Mark Otto, criaram em 2010 o que é hoje o Bootstrap.

O objetivo inicial era otimizar o desenvolvimento através da adoção de uma estrutura única, reduzindo assim, inconsistências entre as diversas formas de codificar que variavam de cada profissional.

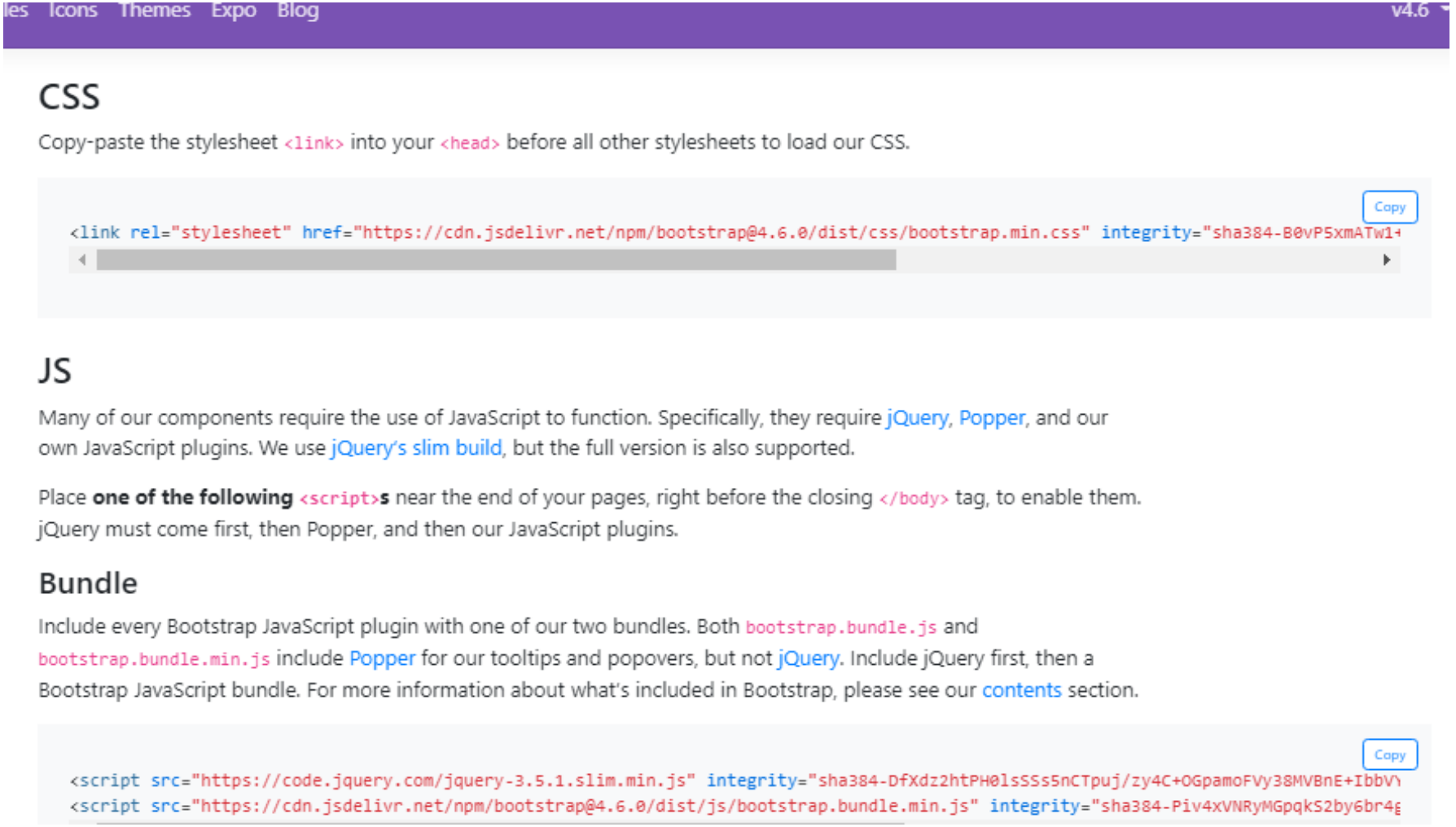
.O Bootstrap oferece uma enorme variedade de Plugins e temas compatíveis com outros frameworks. Além disto, possui integração com qualquer linguagem de programação. Também é um software de código aberto, gratuito. Em pouco tempo após o seu lançamento já recebeu a contribuição de inúmeros desenvolvedores de todo o planeta. Desta forma, tornando-o o software livre mais ativo do mundo.

Como usar?

Para utilizar faça o download do Bootstrap no site **oficial** da ferramenta. Adicione os arquivos “bootstrap.min.css” e “bootstrap.min.js” nas pastas correspondentes a eles dentro do seu site. Depois, é só chamar o CSS e o JS dentro da **tag <head>** do **HTML.** É importante lembrar que este framework acessa a biblioteca jQuery para ativar o comportamento de seus plugins. Portanto, lembre-se de inseri-lo em sua página para que tudo funcione perfeitamente.

https://getbootstrap.com.br/

Uso metodo CDN



CDN Template

-

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css" integrity="sha384-B0vP5xmA

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: jQuery and Bootstrap Bundle (includes Popper) -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+I
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-Piv4xVNRyMGpqkS2by6

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+I
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js" integrity="sha384-+YQ4JLhjyBLPDQt//I+STsc9iw
    -->
  </body>
</html>
```

CDN Template

- Copy

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css" integrity="sha384-B0vP5xmA

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: jQuery and Bootstrap Bundle (includes Popper) -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+I
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-Piv4xVNRyMGpqkS2by6

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+I
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js" integrity="sha384-+YQ4JLhjyBLPDQt//I+STsc9iw
    -->
  </body>
</html>
```

Outros métodos de uso

- **Download bootstrap**

- **Via NPM**

Conteúdo

Diretórios

```
bootstrap/
├── css/
│   ├── bootstrap-grid.css
│   ├── bootstrap-grid.css.map
│   ├── bootstrap-grid.min.css
│   ├── bootstrap-grid.min.css.map
│   ├── bootstrap-reboot.css
│   ├── bootstrap-reboot.css.map
│   ├── bootstrap-reboot.min.css
│   ├── bootstrap-reboot.min.css.map
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   └── bootstrap.min.css.map
└── js/
    ├── bootstrap.bundle.js
    ├── bootstrap.bundle.js.map
    ├── bootstrap.bundle.min.js
    ├── bootstrap.bundle.min.js.map
    ├── bootstrap.js
    ├── bootstrap.js.map
    ├── bootstrap.min.js
    └── bootstrap.min.js.map
```

CSS

Arquivos de folha de estilo do padrão Bootstrap.

JS

Arquivos de JavaScript com os plugins do Bootstrap.

Containers

Os **containers** são o elemento mais básico e são necessários para o sistema de GRID

Containers

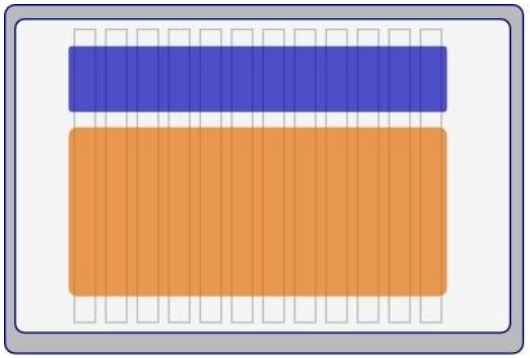
Os **containers** são o elemento mais básico e são necessários para o sistema de GRID

Containers

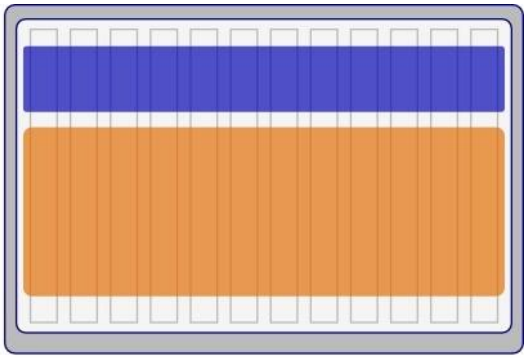
Dois tipo de **container e container-fluid**

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-wid
    <meta http-equiv="X-UA-Compatible" content="ie=
    <title>Document 123</title>
    <link rel="stylesheet" href="node_modules/boots
</head>
<body>

    <div class="container">

        <h1>Hcode Treinamentos</h1>

    </div>

    <div class="container-fluid">

        <h1>Hcode Treinamentos</h1>

    </div>

    <script src="node_modules/jquery/dist/jquery.mi
    <script src="node_modules/bootstrap/dist/js/boo

</body>
```



Layout Fixo
(.container)



Layout Fluido
(.container-fluid)

Mobile first

O Mobile First, como a tradução mesmo sugere, é uma forma de começar pensando primeiro em desenvolvimento para dispositivos móveis, para somente depois evoluir para o desktop. Essa é uma forma de ter uma experiência de excelência no mobile e depois fazer uma adaptação para o desktop

Esse conceito de design responsivo foi criado por Luke Wroblewski

As coisas que tornam os dispositivos móveis únicos têm um grande impacto em como podemos organizar e orientar as pessoas através da experiência gerada pelo nosso conteúdo Web. Apenas diminuir o tamanho de um design baseado em desktop não vai funcionar. Mover seu site para a web móvel faz com que você volte e repense.

.

Responsive

Responsive breakpoints

O CSS é focado por padrão na largura máxima de **576 pixels** e o bootstrap se refere a esse tamanho com a sigla **xs**.

Acima desse tamanho temos a largura mínima de **576 pixels** nomeada **sm**.

Após o tamanho sm temos a largura mínima de **768 pixels** nomeada **md**.

Depois de sm temos a largura mínima de **992 pixels** nomeada **lg**.

Por fim temos a largura mínima de **1200 pixels** nomeada **xl**.

Responsive

```
h1 {
    color: red;
}

@media (min-width: 576px) {
    h1 {
        color: blue;
    }
}

@media (min-width: 768px) {

}

@media (min-width: 992px) {

}

@media (min-width: 1200px) {

}
```

Exercício

Faça uma pagina com um pequeno texto que mude a cor e tipo de letra consoante o seu tamanho.

- **3 breakpoints:**

  defeito: Preto;

  até 581px; azul e arial

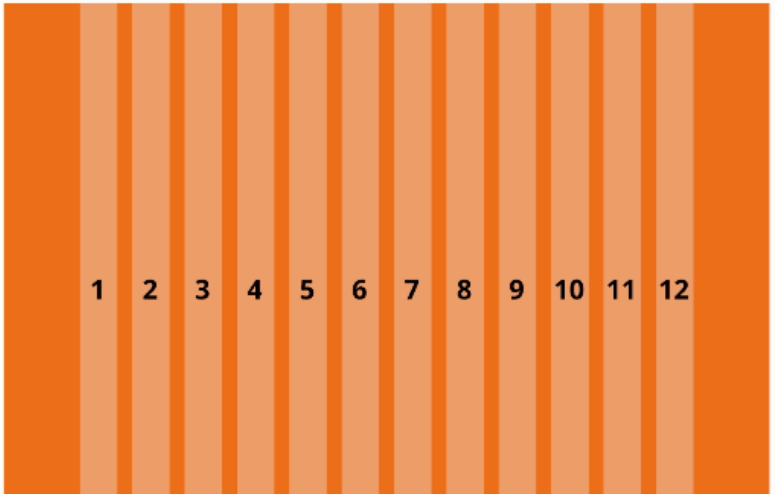
  até 790 px; green e Times new Roman

  até 1100px; vermelho e verdana

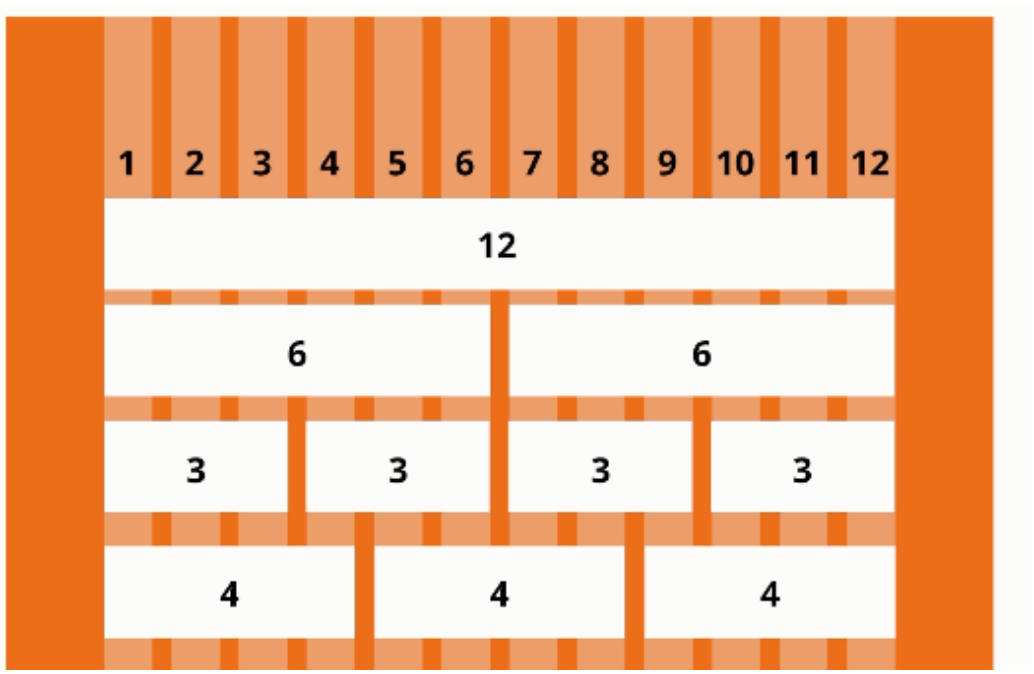
- **Use o bootstrap por CND**

Sistema de Grid

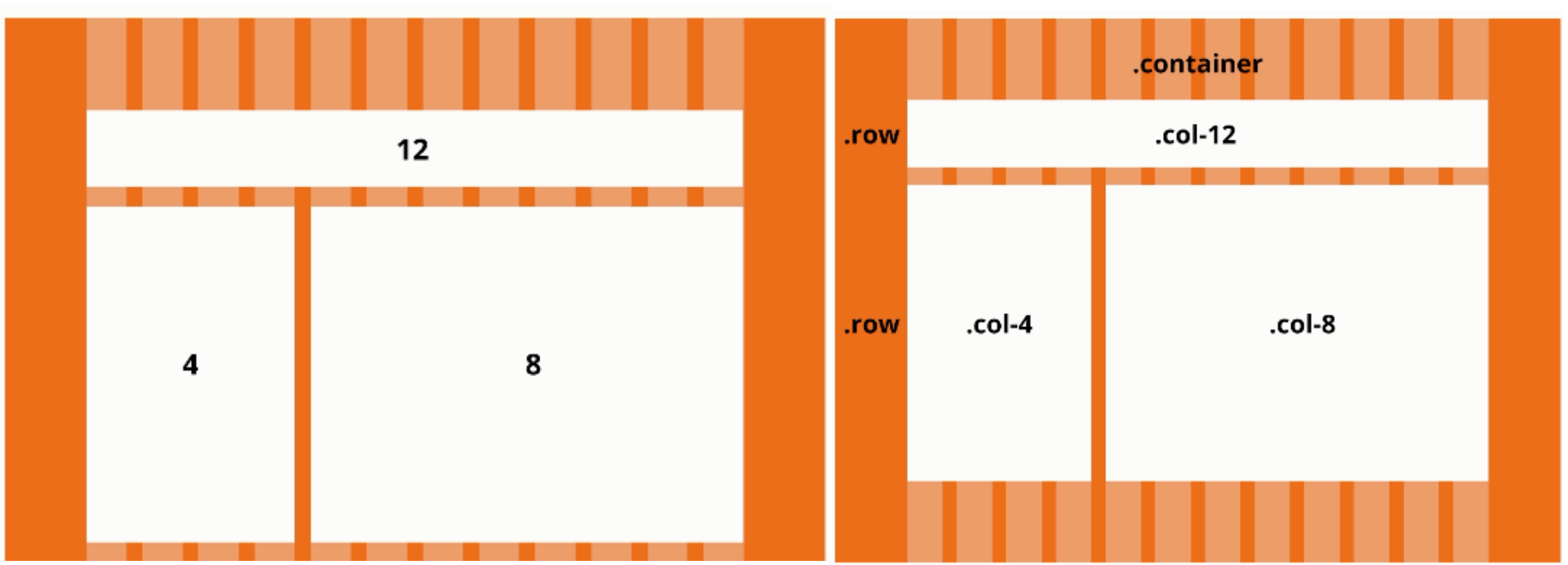
- O sistema de Grid do Bootstrap divide a tela em **12 colunas** e **5 camadas**. Isso faz com que tenhamos de forma prática e poderosa uma ferramenta de criação de telas responsivas.



Estrutura 1



Estrutura 2



Unidades CSS

Relative Lengths

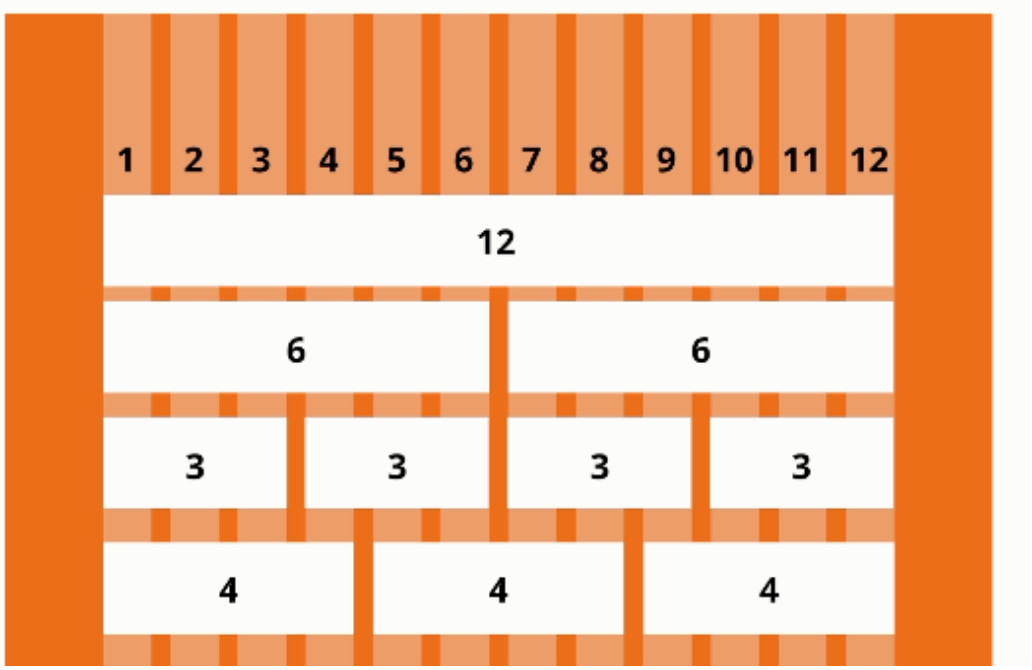
Relative length units specify a length relative to another length property. Relative length units scale better between different rendering medium.

Unit	Description	
em	Relative to the font-size of the element (2em means 2 times the size of the current font)	Try it
ex	Relative to the x-height of the current font (rarely used)	Try it
ch	Relative to the width of the "0" (zero)	Try it
rem	Relative to font-size of the root element	Try it
vw	Relative to 1% of the width of the viewport*	Try it
vh	Relative to 1% of the height of the viewport*	Try it
vmin	Relative to 1% of viewport's* smaller dimension	Try it
vmax	Relative to 1% of viewport's* larger dimension	Try it
%	Relative to the parent element	Try it

Exercício

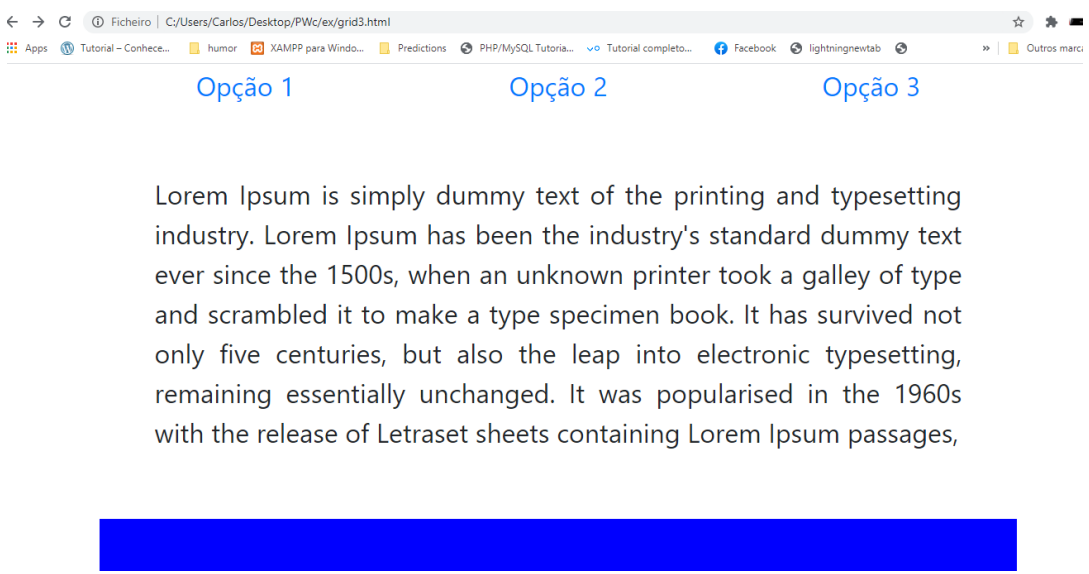
Faça este layout e com cores diferentes para cada row

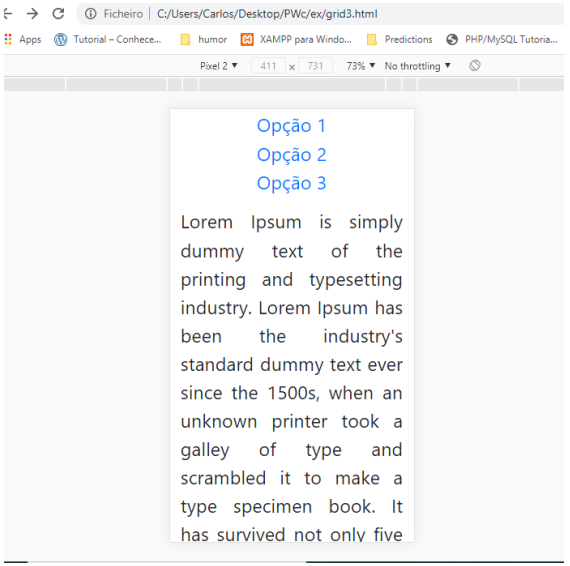
.



Exercício

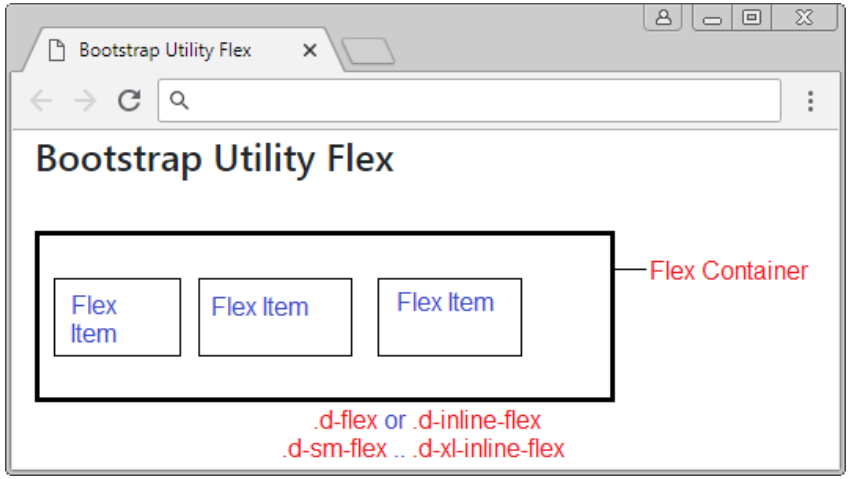
Faça um simples layout com um menu com 3 opções e o respectivo content mudará consoante o link



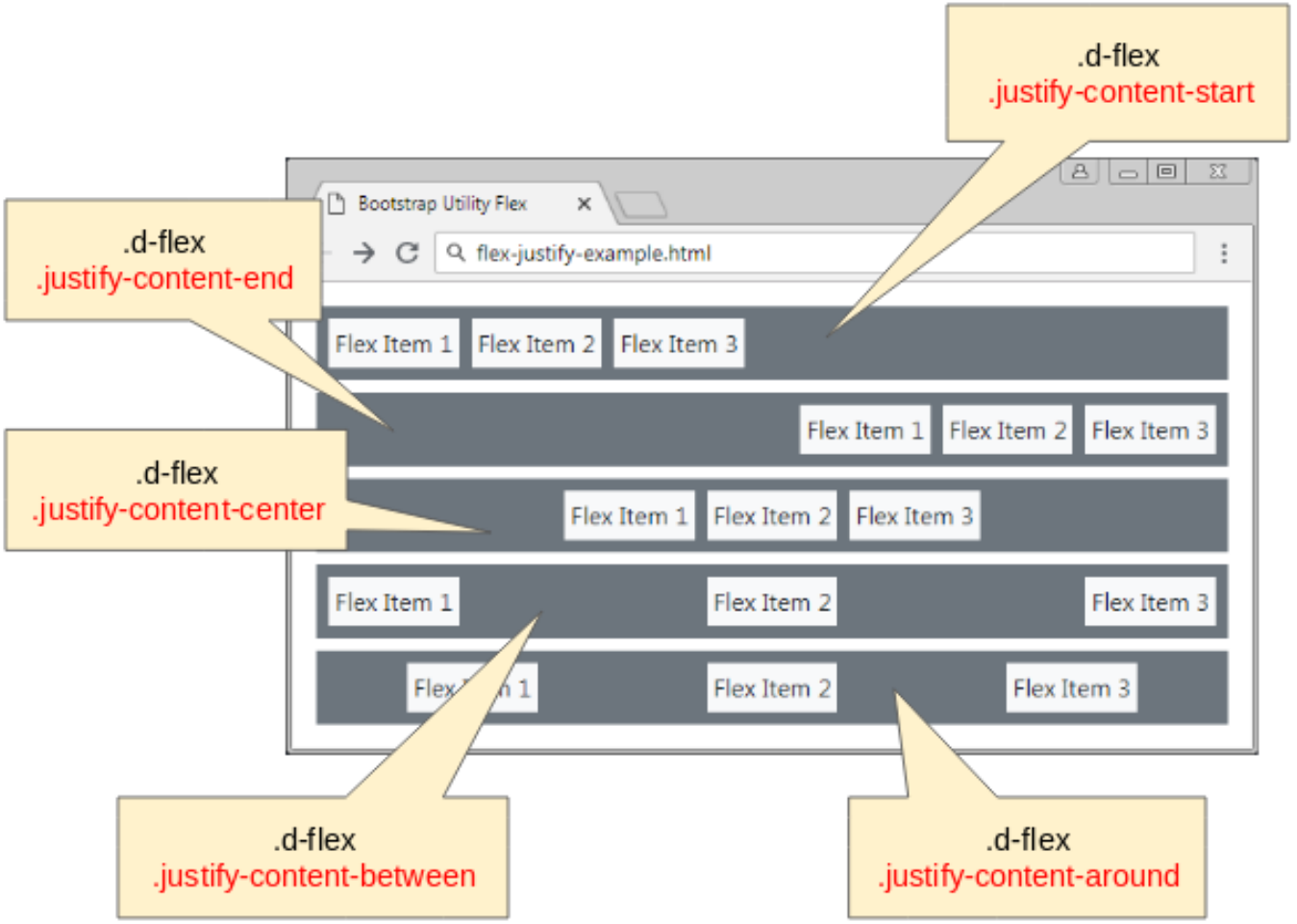


FLEXBOX

Por um longo tempo, as únicas ferramentas compatíveis entre browsers disponíveis para criação de layouts CSS eram coisas como floats e posicionamento. Estas são boas e funcionam, mas em alguns casos também são limitadas e frustrantes..



Flexbox


.d-flex
.justify-content-start
.d-flex
.justify-content-end
.d-flex
.justify-content-center
.d-flex
.justify-content-between
.d-flex
.justify-content-around
Bootstrap Utility Flex
flex-justify-example.html
Flex Item 1
Flex Item 2
Flex Item 3

Tipografia

h1. Bootstrap heading

h2. Bootstrap heading

h3. Bootstrap heading

h4. Bootstrap heading

h5. Bootstrap heading

h6. Bootstrap heading

```
<p class="h1">h1. Bootstrap heading</p>
<p class="h2">h2. Bootstrap heading</p>
<p class="h3">h3. Bootstrap heading</p>
<p class="h4">h4. Bootstrap heading</p>
<p class="h5">h5. Bootstrap heading</p>
<p class="h6">h6. Bootstrap heading</p>
```

Inline

Remove a list's bullets and apply some light `margin` with a combination of two classes, `.list-inline` and `.list-inline-item`.

This is a list item. And another one. But they're displayed inline.

Copy

```
<ul class="list-inline">
  <li class="list-inline-item">This is a list item.</li>
  <li class="list-inline-item">And another one.</li>
  <li class="list-inline-item">But they're displayed inline.</li>
</ul>
```

Tipografia

Color

.text-primary

.text-secondary

.text-success

.text-danger

.text-warning

.text-info

.text-light

.text-dark

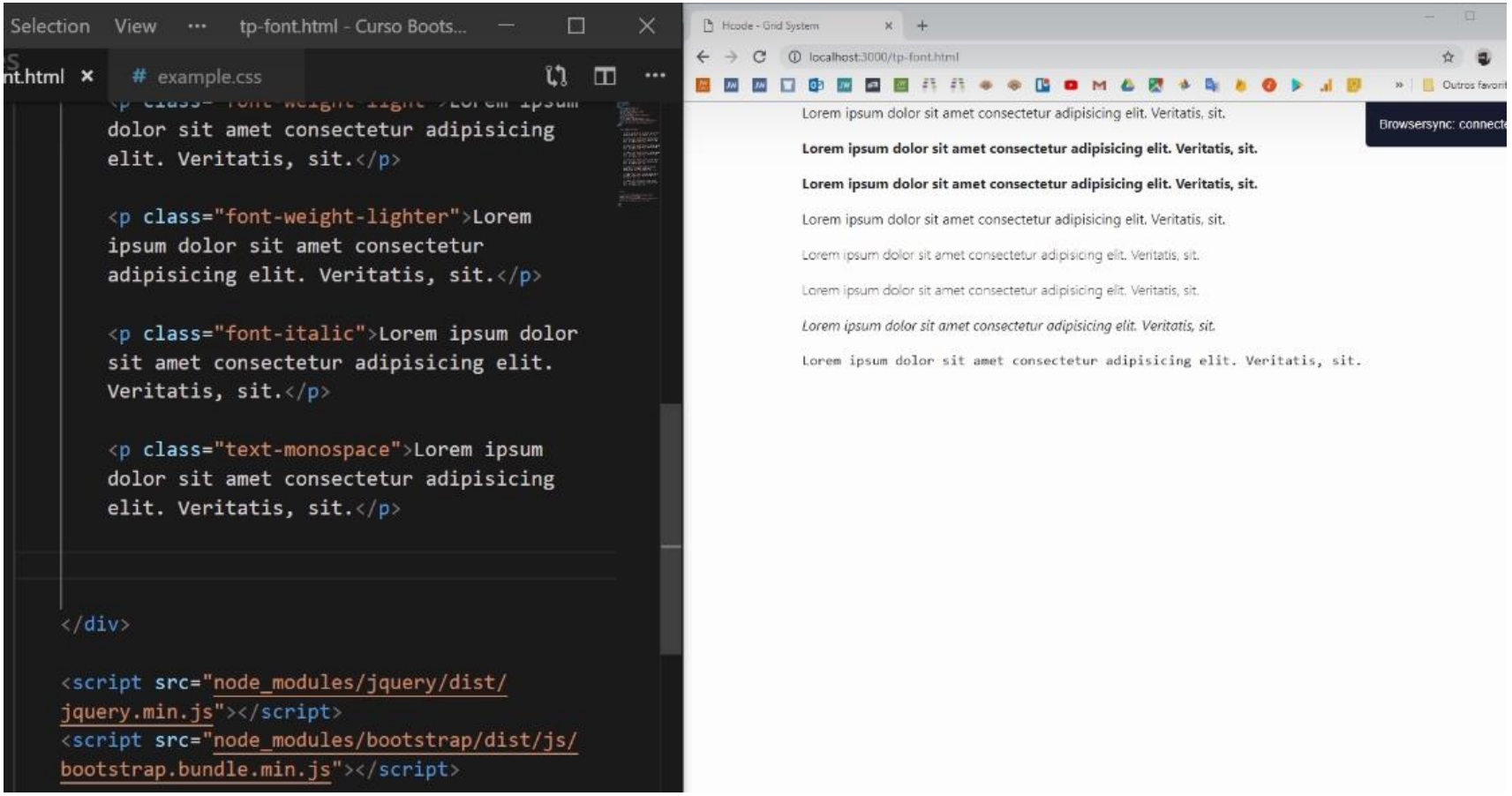
.text-muted

.text-white

Copy

```
<p class="text-primary">.text-primary</p>
<p class="text-secondary">.text-secondary</p>
<p class="text-success">.text-success</p>
<p class="text-danger">.text-danger</p>
<p class="text-warning">.text-warning</p>
<p class="text-info">.text-info</p>
<p class="text-light bg-dark">.text-light</p>
<p class="text-dark">.text-dark</p>
<p class="text-muted">.text-muted</p>
<p class="text-white bg-dark">.text-white</p>
```

Tipografia


Selection View ··· tp-font.html - Curso Boots...
example.css
dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.

<script src="node_modules/jquery/dist/jquery.min.js"></script>
<script src="node_modules/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
Hcode - Grid System
localhost:3000/tp-font.html
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Browsersync: connecte
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis, sit.

Tipografia: classes

More Typography Classes

The Bootstrap 4 classes below can be added to style HTML elements further:

Class	Description	Example
.font-weight-bold	Bold text	Try it
.font-weight-bolder	Bolder text	Try it
.font-italic	Italic text	Try it
.font-weight-light	Light weight text	Try it
.font-weight-lighter	Lighter weight text	Try it
.font-weight-normal	Normal text	Try it
.lead	Makes a paragraph stand out	Try it
.small	Indicates smaller text (set to 80% of the size of the parent)	Try it
.text-left	Indicates left-aligned text	Try it
.text-*-left	Indicates left-aligned text on small, medium, large or xlarge screens	Try it
.text-break	Prevents long text from breaking layout	Try it
.text-center	Indicates center-aligned text	Try it
.text-*-center	Indicates center-aligned text on small, medium, large or xlarge screens	Try it
	underline from a link	Try it

.text-right	Indicates right-aligned text	Try it
.text-*-right	Indicates right-aligned text on small, medium, large or xlarge screens	Try it
.text-justify	Indicates justified text	Try it
.text-monospace	Monospaced text	Try it
.text-nowrap	Indicates no wrap text	Try it
.text-lowercase	Indicates lowercased text	Try it
.text-reset	Resets the color of a text or a link (inherits the color from its parent)	Try it
.text-uppercase	Indicates uppercased text	Try it
.text-capitalize	Indicates capitalized text	Try it
.initialism	Displays the text inside an <abbr> element in a slightly smaller font size	Try it
.list-unstyled	Removes the default list-style and left margin on list items (works on both and). This class only applies to immediate children list items (to remove the default list-style from any nested lists, apply this class to any nested lists as well)	Try it
.list-inline	Places all list items on a single line (used together with .list-inline-item on each elements)	Try it
.pre-scrollable	Makes a <pre> element scrollable	Try it

Imagens

In Bootstrap 4 add `.img-fluid` **class to make the image responsive.**

Example: `<img src="https://example-image.com" class="img-fluid">`

The gray border of each picture shows the size of his parent element.

Detailed documentation and more examples about Bootstrap image slider you can find in our Bootstrap Images Docs

Tables

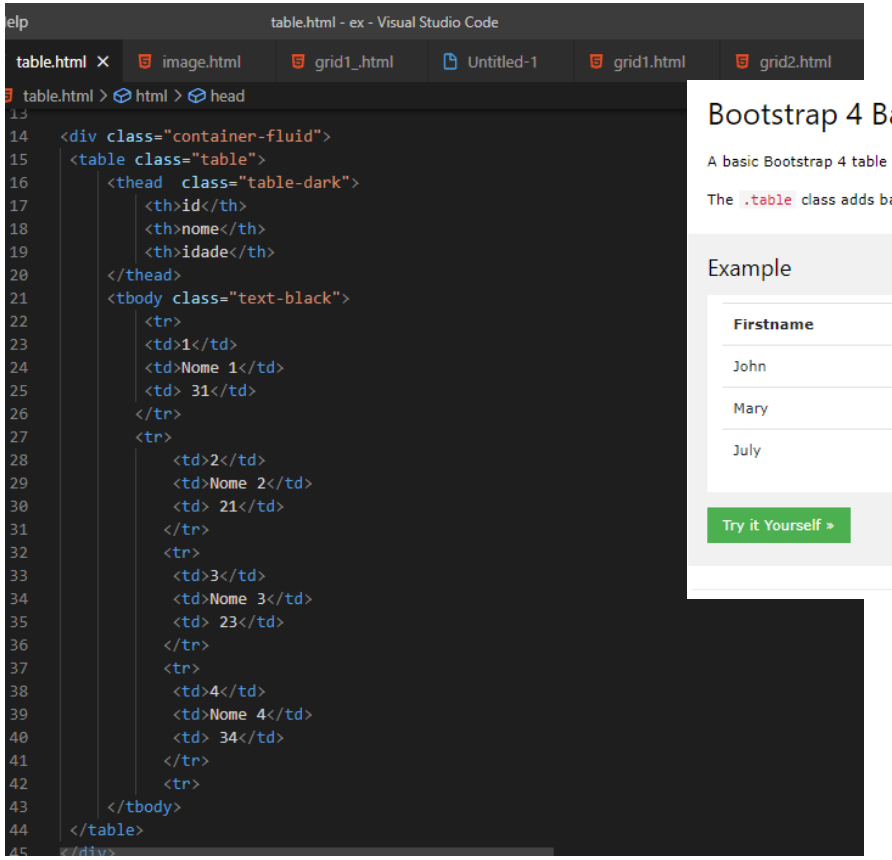
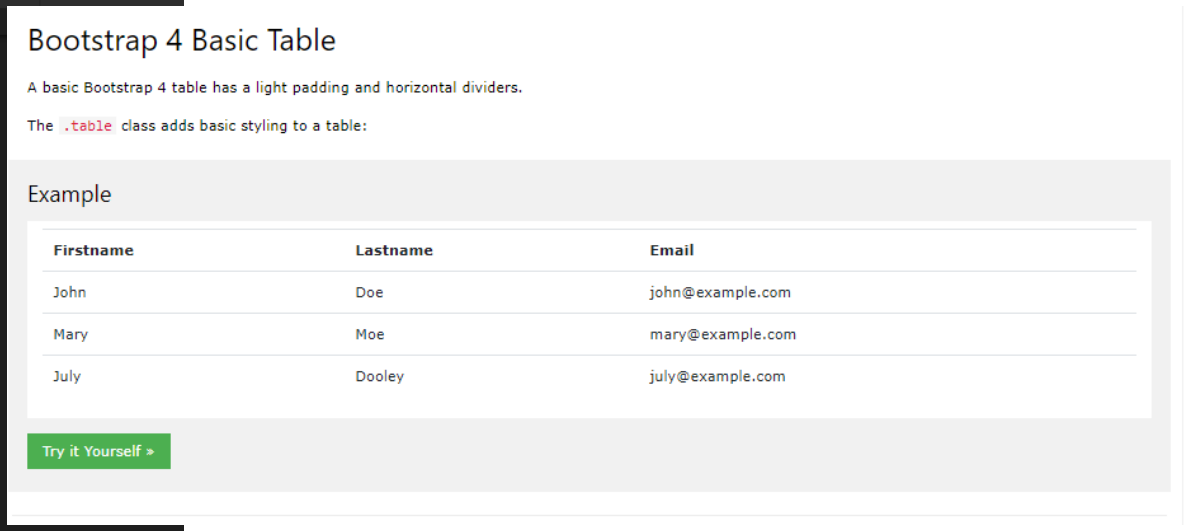


table.html - ex - Visual Studio Code
table.html
image.html
grid1_.html
Untitled-1
grid1.html
grid2.html
table.html > html > head

<table class="table">
<thead class="table-dark">
<th>id</th>
<th>nome</th>
<th>idade</th>
</thead>
<tbody class="text-black">
<tr>
<td>1</td>
<td>Nome 1</td>
<td> 31</td>
</tr>
<tr>
<td>2</td>
<td>Nome 2</td>
<td> 21</td>
</tr>
<tr>
<td>3</td>
<td>Nome 3</td>
<td> 23</td>
</tr>
<tr>
<td>4</td>
<td>Nome 4</td>
<td> 34</td>
</tr>
<tr>
</tbody>
</table>

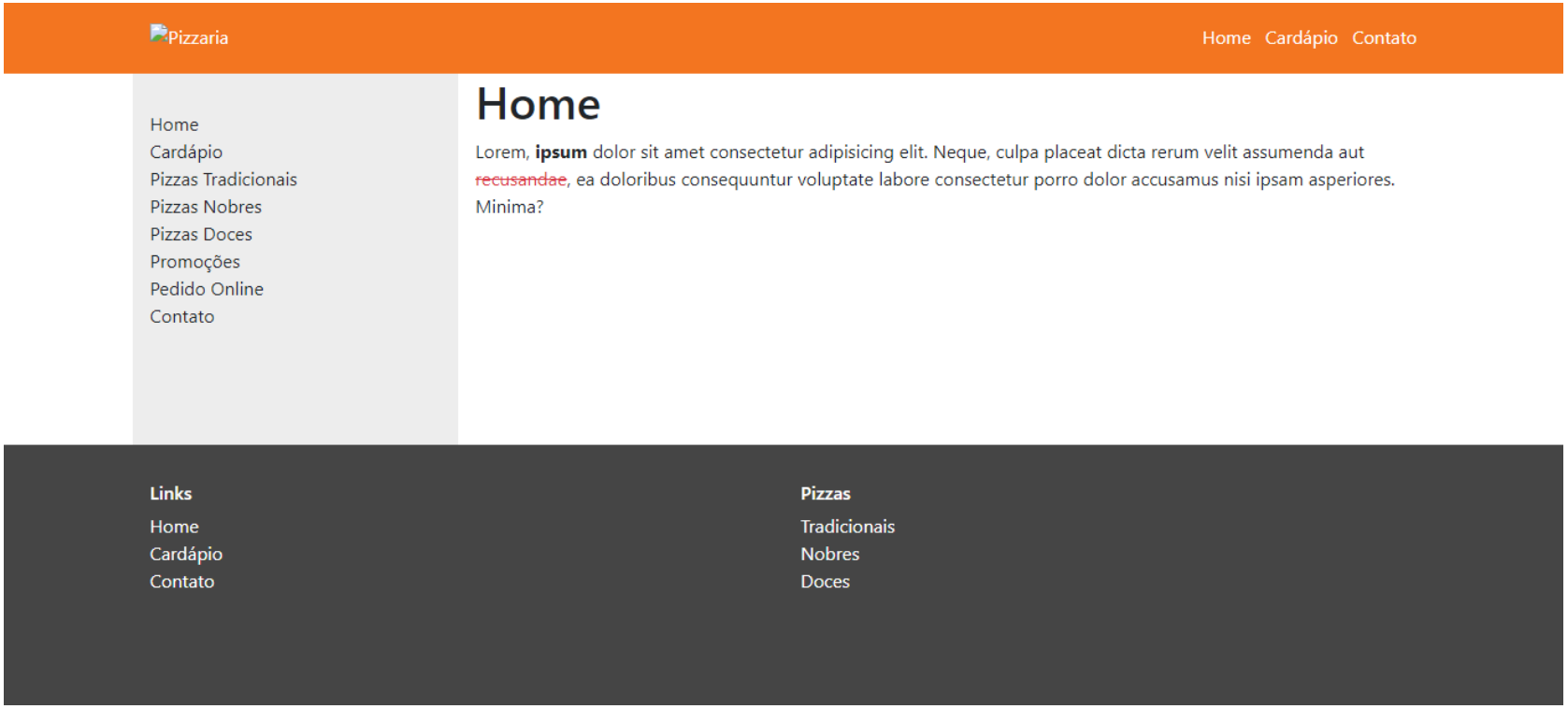

Bootstrap 4 Basic Table
A basic Bootstrap 4 table has a light padding and horizontal dividers.
The .table class adds basic styling to a table:
Example
Firstname
Lastname
Email
John
Doe
john@example.com
Mary
Moe
mary@example.com
July
Dooley
july@example.com
Try it Yourself »

Carousel


```
<div id="carouselExampleControls" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
    <div class="carousel-item">
      <img src="..." class="d-block w-100" alt="...">
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleControls" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleControls" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

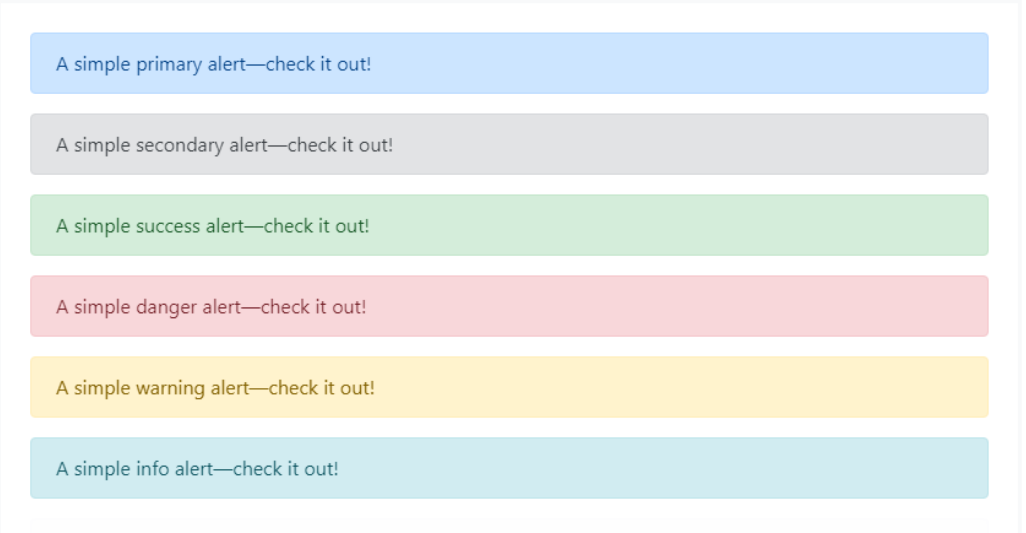
Exercício

Usa o bootstrap para fazer este layout com o posicionamento da figura abaixo. Faça 3 links funcionais: 1-home texto **2-cardápio** fotos **Promoções-** tabela



Alerts

```
<div class="alert alert-primary" role="alert">
  This is a primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  This is a secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  This is a success alert—check it out!
</div>
<div class="alert alert-danger" role="alert">
  This is a danger alert—check it out!
</div>
<div class="alert alert-warning" role="alert">
  This is a warning alert—check it out!
</div>
<div class="alert alert-info" role="alert">
  This is a info alert—check it out!
</div>
<div class="alert alert-light" role="alert">
  This is a light alert—check it out!
</div>
<div class="alert alert-dark" role="alert">
  This is a dark alert—check it out!
```

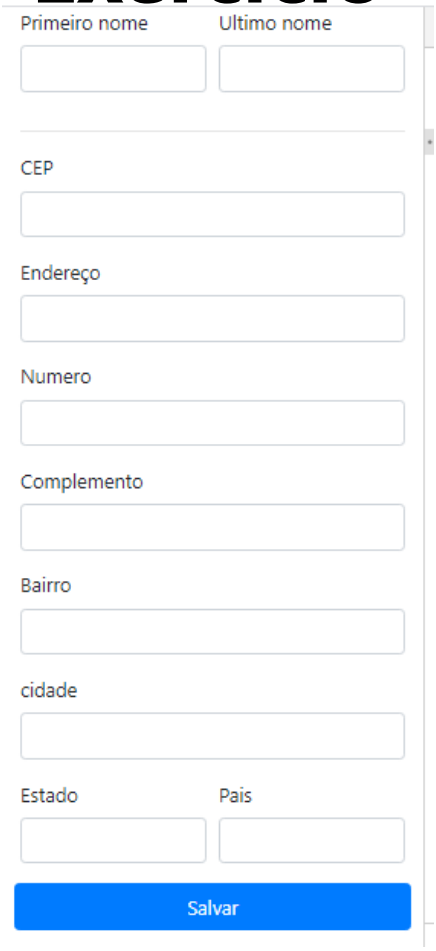


buttons

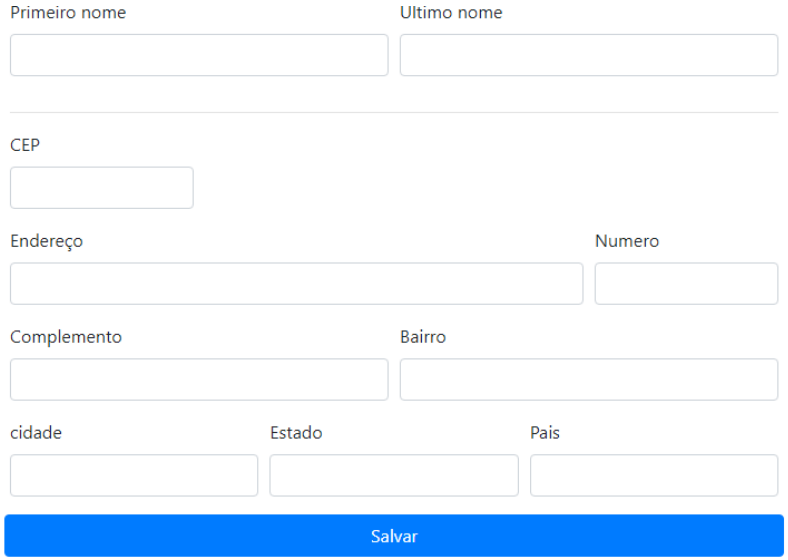
Copy

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>
```

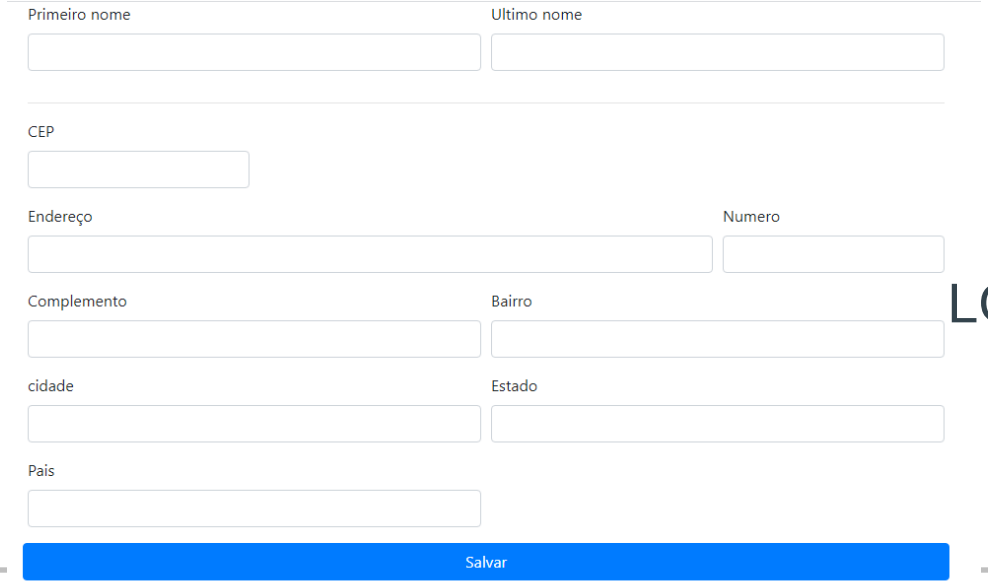
Exercício



Pequeno



SM



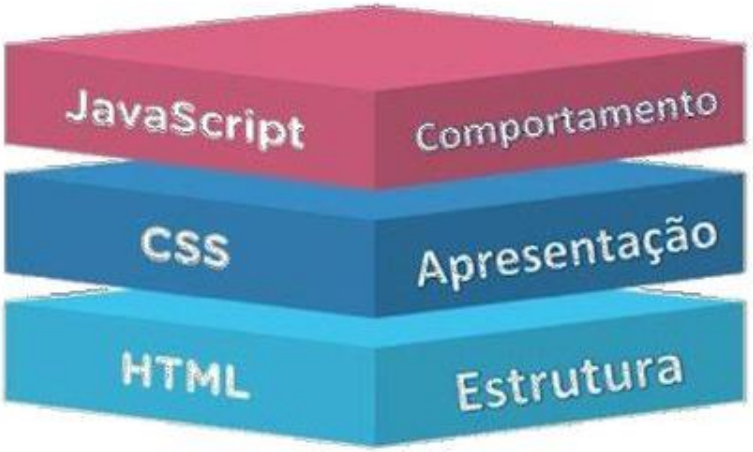
LG

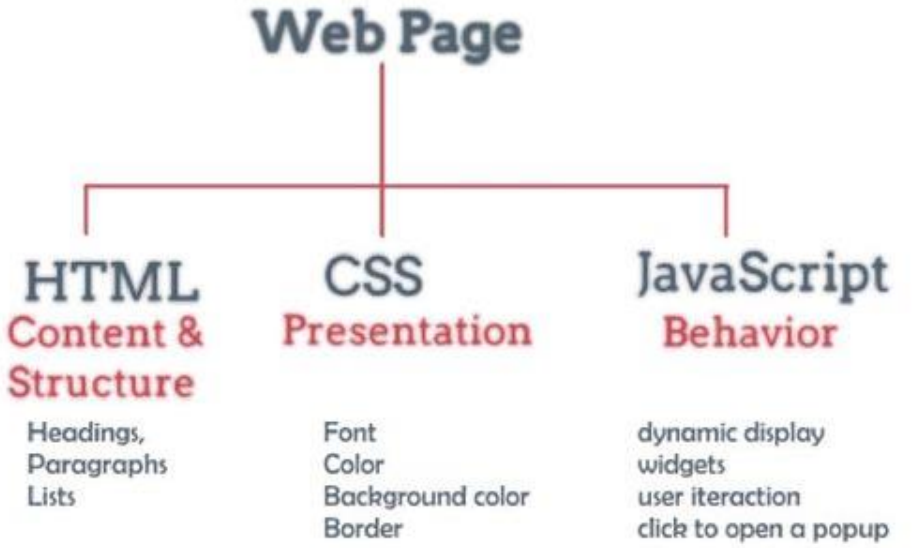
Javascript

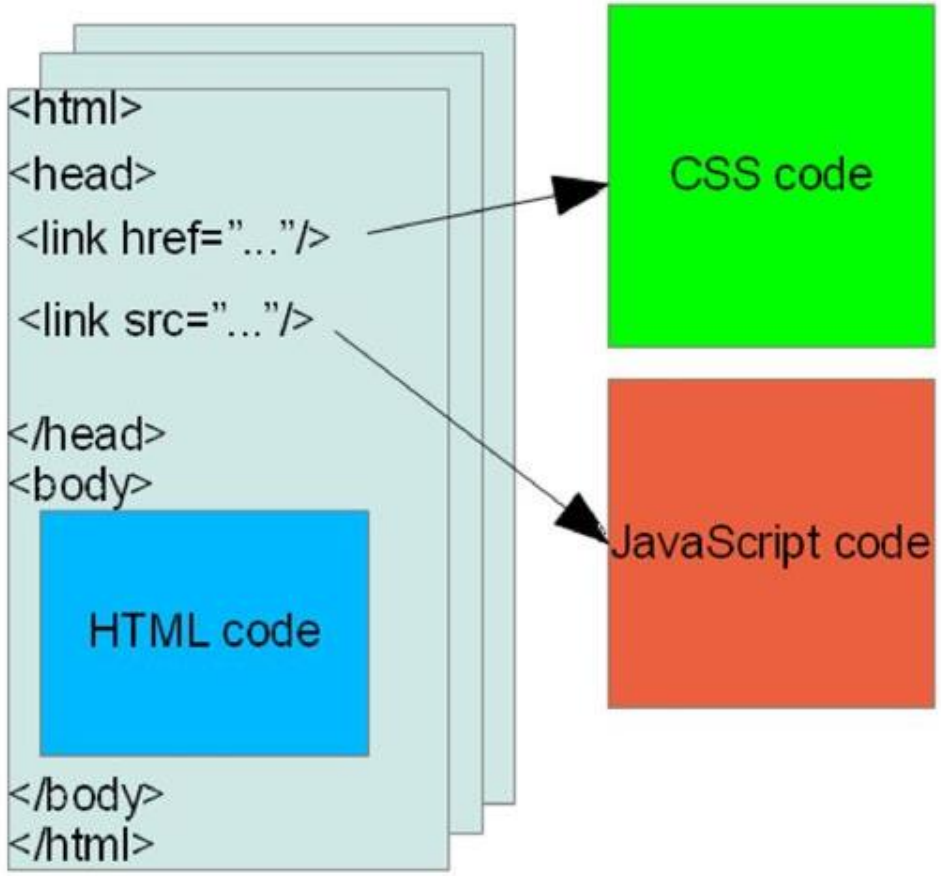
Norberto Moreira

- *Script* é um conjunto de instruções em código.
  - É uma linguagem de programação que executa diversas funções no interior de um programa de computador.
  - **Linguagem de *script*** são linguagens de programação executadas no interior de programas e/ou de outras linguagens de programação e que não estão restritas a um determinado ambiente.
  - As linguagens de *script* servem para estender a funcionalidade de um programa e/ou controlá-lo, acedendo à sua API e, são frequentemente usadas como ferramentas de configuração e instalação em sistemas operacionais (Shell script).

Java	JavaScript
Linguagem de **programação** orientada a objetos;	**Linguagem** de **script** orientada a objetos;
Cria aplicações que são executados em uma máquina virtual ou browser;	Executado apenas em um browser.
Precisa ser compilado;	Não precisa de compilador
É autônomo;	Não cria aplicativos autônomos
	Está dentro do HTML e fornece níveis de interatividade para páginas da web


JavaScript
Comportamento
CSS
Apresentação
HTML
Estrutura


Web Page
HTML
Content & Structure
Headings,
Paragraphs
Lists
CSS
Presentation
Font
Color
Background color
Border
JavaScript
Behavior
dynamic display
widgets
user iteraction
click to open a popup


<html>
<head>
<link href="..."/>
<link src="..."/>
</head>
<body>
HTML code
</body>
</html>
CSS code
JavaScript code

JavaScript é Case Sensitive

Case Sensitive significa algo como “sensível à caixa das letras” ou “sensível a maiúsculas e minúsculas”.

> *Exemplo:*

`const Fruta = "Banana"` é diferente de `const fruta = "Maçã"`

O JavaScript identifica que é outra variável, bastando um caractere da declaração ser diferente, no caso **F** é diferente de **f**, com isso, o JS interpreta como uma variável diferente.

Onde colocar ?

- Na tag <script> do HTML (pode ser no head ou no body)

```
<script>
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

- Colocar num ficheiro próprio .js
- Fazer o *import* - Adicionar o link do ficheiro externo
  1. Colocar no head (lê logo no início)
  2. Colocar antes do </body> (lê após carregar a página – opção mais rápida e a ideal quando temos código dependente de objetos HTML)

exemplo

- Botão que reage quando clicado pelo utilizador

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript example</title>
    <script type="text/javascript" src="please.js"></script>
</head>
<body>
    <div>
        <button onclick="sayMagicWord();">Say the Magic Word</button>
    </div>
</body>
</html>
```

```
function sayMagicWord() {
    alert("PLEASE!");
}
```

Display

JavaScript Output

‹ Previous

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using `innerHTML`.
- Writing into the HTML output using `document.write()`.
- Writing into an alert box, using `window.alert()`.
- Writing into the browser console, using `console.log()`.

Using innerHTML

Para aceder a um element do HTML, JavaScript pode usar document.getElementById(id) method. O id identifica o elemento html. O innerHTML propriedade define o content do html:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Using document.write()

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Using window.alert()

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

Using console.log()

Podemos chamar o console.log() no browser para mostrar/ debug do no script

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

Variáveis javascript

Declarações

Existem três tipos de declarações em JavaScript.

`var`
Declara uma variável, opcionalmente, inicializando-a com um valor.

`let`
Declara uma variável local de escopo do bloco, opcionalmente, inicializando-a com um valor.

`const`
Declara uma constante de escopo de bloco, apenas de leitura.

Declarando variáveis

Você pode declarar uma variável de três formas:

- Com a palavra chave `var`. Por exemplo, `var x = 42`. Esta sintaxe pode ser usada para declarar tanto variáveis locais como variáveis globais.
- Por simples adição de valor. Por exemplo, `x = 42`. Isso declara uma variável global. Essa declaração gera um aviso de advertência no JavaScript. Você não deve usar essa variante.
- Com a palavra chave `let`. Por exemplo, `let y = 13`. Essa sintaxe pode ser usada para declarar uma variável local de escopo de bloco. Veja escopo de variável abaixo.

Variáveis tipos

Descrição

Esta tabela resume os possíveis valores que são retornados pelo `typeof`:

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

Example

The following code shows how to implement **typeof** operator.

Objeto

```
var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

Operadores matemáticos

Operador	Descrição	Exemplo	Resultado
+	Adição	x = 5 + 2	7
-	Subtração	x = 5 – 2	3
*	Multiplicação	x = 5 * 2	10
/	Divisão	x = 5 / 2	2,5
%	Resto da divisão	x = 5 % 2	1
++	Incremento	x = ++6 x = 6++	7
--	Decremento	x = --4 x = 4--	3

*Os operadores aritméticos são utilizados para realizar operações numéricas com os dados utilizados pelo programa.

Operadores atribuição

Operador	Exemplo	Sinônimo	Resultado
=	x = y	x = y	5
+=	x += y	x = x + y	15
-=	x -= y	x = x - y	5
*=	x *= y	x = x * y	30
/=	x /= y	x = x / y	2
%=	x %= y	x = x % y	0

Operadores Relacionais

OPERADOR	DESCRIÇÃO
==	Igual a
<> ou !=	Diferente de
===	Igual e do mesmo tipo que
!==	Valor e tipos diferentes de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

*Situações onde é necessário comparar informações para que o programa possa tomar uma decisão.

Operadores Lógicos

OPERADOR	DESCRIÇÃO
! NOT	Negação
&& AND	Conjunção
ll OR	Disjunção

* Usados para momentos onde se é necessário trabalhar com duas ou mais condições

If…Else

```
if (expression) {
   Statement(s) to be executed if expression is true
} else {
   Statement(s) to be executed if expression is false
}
```

Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the 'if' block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var age = 15;

            if( age > 18 ) {
               document.write("<b>Qualifies for driving</b>");
            } else {
               document.write("<b>Does not qualify for driving</b>");
            }
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

Switch

```
switch (expression) {
   case condition 1: statement(s)
   break;

   case condition 2: statement(s)
   break;
   ...

   case condition n: statement(s)
   break;

   default: statement(s)
}
```

The **break** statements indicate the end of a particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

Live Demo

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var grade = 'A';
            document.write("Entering switch block<br />");
            switch (grade) {
               case 'A': document.write("Good job<br />");
               break;

               case 'B': document.write("Pretty good<br />");
               break;

               case 'C': document.write("Passed<br />");
               break;

               case 'D': document.write("Not so good<br />");
               break;

               case 'F': document.write("Failed<br />");
               break;

               default:  document.write("Unknown grade<br />")
            }
            document.write("Exiting switch block");
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

While loop

The syntax of **while loop** in JavaScript is as follows −

```
while (expression) {
   Statement(s) to be executed if expression is true
}
```

Example

Try the following example to implement while loop.

Live Demo

```
<html>
   <body>

      <script type = "text/javascript">
         <!--
            var count = 0;
            document.write("Starting Loop ");

            while (count < 10) {
               document.write("Current Count : " + count + "<br />");
               count++;
            }

            document.write("Loop stopped!");
         //-->
      </script>
```

Do… while

```
do {
   Statement(s) to be executed;
} while (expression);
```

Note – Don't miss the semicolon used at the end of the **do...while** loop.

Example

Try the following example to learn how to implement a **do-while** loop in JavaScript.

Live Demo

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var count = 0;

            document.write("Starting Loop" + "<br />");
            do {
               document.write("Current Count : " + count + "<br />");
               count++;
            }

            while (count < 5);
            document.write ("Loop stopped!");
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

For...loop

```
for (initialization; test condition; iteration statement) {
   Statement(s) to be executed if test condition is true
}
```

Example

Try the following example to learn how a **for** loop works in JavaScript.

Live Demo

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var count;
            document.write("Starting Loop" + "<br />");

            for(count = 0; count < 10; count++) {
               document.write("Current Count : " + count );
               document.write("<br />");
            }
            document.write("Loop stopped!");
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

For...in

```
for (variablename in object) {
   statement or block to execute
}
```

In each iteration, one property from **object** is assigned to **variablename** and this loop continues till all the properties of the object are exhausted.

Example

Try the following example to implement 'for-in' loop. It prints the web browser's **Navigator** object.

Live Demo

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var aProperty;
            document.write("Navigator Object Properties<br /> ");
            for (aProperty in navigator) {
               document.write(aProperty);
               document.write("<br />");
            }
            document.write ("Exiting from the loop!");
         //-->
      </script>
      <p>Set the variable to different object and then try...</p>
   </body>
</html>
```

Arrays

Declaration

There are two syntaxes for creating an empty array:

```
let arr = new Array();
let arr = [];
```

Almost all the time, the second syntax is used. We can supply initial elements in the brackets:

```
let fruits = ["Apple", "Orange", "Plum"];
```

Array elements are numbered, starting with zero.

We can get an element by its number in square brackets:

```
let fruits = ["Apple", "Orange", "Plum"];

alert( fruits[0] ); // Apple
alert( fruits[1] ); // Orange
alert( fruits[2] ); // Plum
```

We can replace an element:

Arrays

```
1  let fruits = ["Apple", "Orange", "Plum"];
2
3  alert( fruits.length ); // 3
```

Arrays

Array [9,5,64,6,9,1,4,5,6]

- **Dar a soma de todos os elementos**
- **Dar a soma só dos pares**
- **Imprimir no console.log os elementos**
- **Substituir o os menor de 9 por 0 e imprimir com o document.write o vector**

Arrays de objectos

```
yourArray.forEach(function (arrayItem) {
    var x = arrayItem.prop1 + 2;
    console.log(x);
});
```

```
let cars = [
  {
    "color": "purple",
    "type": "minivan",
    "registration": new Date('2017-01-03'),
    "capacity": 7
  },
  {
    "color": "red",
    "type": "station wagon",
    "registration": new Date('2018-03-03'),
    "capacity": 5
  },
  {
    ...
  },
  ...
]
```

Pessoa = [{ nome:pessoa1, idade=21,cidade:Porto},

{ nome:pessoa2, idade=25,cidade:Maia},]]

- **Imprimir array de objectos**
- **Imprimir so que m vive no Porto**
- **Quem vive na Maia mudar para idade para 21**

Arrays Pop,push,Shift,unshift

`pop`

Extracts the last element of the array and returns it:

```
let fruits = ["Apple", "Orange", "Pear"];

alert( fruits.pop() ); // remove "Pear" and alert it

alert( fruits ); // Apple, Orange
```

`push`

Append the element to the end of the array:

```
let fruits = ["Apple", "Orange"];

fruits.push("Pear");

alert( fruits ); // Apple, Orange, Pear
```

`shift`

Extracts the first element of the array and returns it:

```
let fruits = ["Apple", "Orange", "Pear"];

alert( fruits.shift() ); // remove Apple and alert it

alert( fruits ); // Orange, Pear
```

`unshift`

Add the element to the beginning of the array:

```
let fruits = ["Orange", "Pear"];

fruits.unshift('Apple');

alert( fruits ); // Apple, Orange, Pear
```

Funções

JavaScript Function Syntax

A JavaScript function is defined with the `function` keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

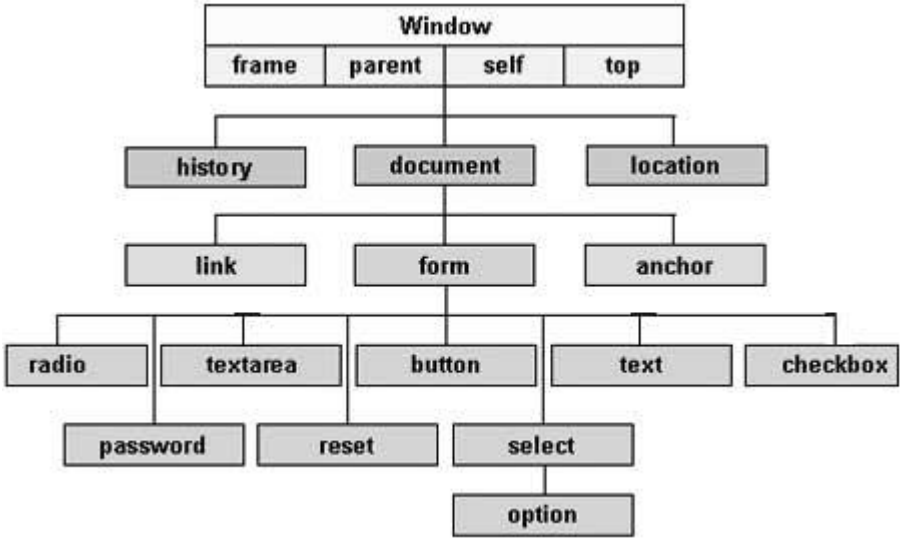
The parentheses may include parameter names separated by commas:
(*parameter1, parameter2, ...*)

The code to be executed, by the function, is placed inside curly brackets: **{}**

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Introdução ao DOM

O Document Object Model ou simplesmente DOM é utilizado pelo navegador Web para representar a sua página Web. Quando altera-se esse modelo com o uso do **Javascript** altera-se também a página Web. É muito mais fácil trabalhar com DOM do que diretamente com código **HTML** ou **CSS**. Um dos grandes responsáveis por isso tudo é o objeto document que é responsável por conceder ao código Javascript todo o acesso a **árvore DOM** do navegador Web. Portanto, qualquer coisa criado pelo navegador Web no modelo da página Web poderá ser acessado através do objeto Javascript



DOM propriedades InnerText e InnerHtml

O `innerText` altera o conteúdo de um elemento de sua página (DOM) com o conteúdo tratado apenas como texto. Por exemplo:

```
document.getElementById('Teste').innerText = '<b>teste</b>'
```

Irá exibir:

teste

Já o `innerHTML` altera o conteúdo de um elemento com o conteúdo tratado como HTML.

Por exemplo, este código:

```
document.getElementById('Teste').innerHTML = '<b>teste</b>'
```

Será exibido dessa maneira:

teste

DOM Finding elements

Metodos

- HTML elements by id
- HTML elements by tag name
- HTML elements by class name
- Finding HTML elements by CSS selectors
- HTML elements by HTML object collections

Eventos

No caso da web, eventos são disparados dentro da janela do navegador, e tende a estarem anexados a algum item especifico nele — pode ser um único elemento, um conjunto de elementos, o HTML carregado na guia atual, ou toda a janela do navegador. Existem vários tipos diferentes de eventos que podem vir a acontecer, por exemplo:

- O utilizador clicou com o mouse sobre um certo elemento ou passando o cursor do mouse sobre um certo elemento.
- O utilizador pressionando uma tecla do teclado.
- O utilizador redimensionando ou fechando a janela do navegador.
- Uma pagina da web terminando de carregar.
- Um formulário sendo enviado.
- Um vídeo sendo reproduzido, interrompido, ou terminando sua reprodução.
- Um erro ocorrendo.

Eventos

No caso da web, eventos são disparados dentro da janela do navegador, e tende a estarem anexados a algum item especifico nele — pode ser um único elemento, um conjunto de elementos, o HTML carregado na guia atual, ou toda a janela do navegador. Existem vários tipos diferentes de eventos que podem vir a acontecer, por exemplo:

- O utilizador clicou com o mouse sobre um certo elemento ou passando o cursor do mouse sobre um certo elemento.
- O utilizador pressionando uma tecla do teclado.
- O utilizador redimensionando ou fechando a janela do navegador.
- Uma pagina da web terminando de carregar.
- Um formulário sendo enviado.
- Um vídeo sendo reproduzido, interrompido, ou terminando sua reprodução.
- Um erro ocorrendo.

Eventos exemplos

.

onBlur	remove o foco do elemento
onChange	muda o valor do elemento
onClick	o elemento é clicado pelo usuário
onFocus	o elemento é focado
onKeyPress	o usuário pressiona uma tecla sobre o elemento
onLoad	carrega o elemento por completo
onMouseOver	define ação quando o usuário passa o mouse sobre o elemento
onMouseOut	define ação quando o usuário retira o mouse sobre o elemento
onSubmit	define ação ao enviar um formulário

Como usar os eventos em JavaScript

Existem diversas maneiras de se aplicar esses eventos aos elementos **HTML**, são elas:

- Inline
- Em um arquivo externo, usando um manipulador de eventos

.

EVENT inline

Vamos ver um exemplo de como usar um evento de maneira inline em JavaScript. Dessa forma é preciso que tudo seja definido diretamente na tag do elemento, dessa forma:

.

```
<!DOCTYPE html>
<html>
<head>
<title>Usando o evento onClick no Javascript</title>
</head>
<body>

<h1 onclick="this.innerHTML='Isso acontece quando usamos o evento onClick!'"
>Clique nesse link para testar o evento onClick!</h1>

</body>
</html>
```