

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 1. Preparação

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 1.1. Visão geral do módulo

Prof. Danilo Ferreira E Silva

Nesta aula



- ☐ Apresentar uma visão geral do módulo.
- ☐ Entender a importância de se estudar fundamentos do desenvolvimento Front end.

Fundamentos



- ❑ HTML
- ❑ CSS
- ❑ JavaScript

Por que estudar fundamentos?



- ☐ Necessário para entender a plataforma Web.
- ☐ Nem sempre usamos frameworks como React, Angular, Vue, etc.
- ☐ Mesmo usando um framework, é necessário entender HTML, CSS e, principalmente, JavaScript.

O que vamos precisar



- ☐ Servidor Web (**Node.js + live-server**)
- ☐ Navegador (**Google Chrome**)
- ☐ Editor de código fonte (**VSCode**)

Próxima aula



- ❑ Instalação do Node.js e pacote live-server.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 1.2. Instalação do Node.js e live-server

Prof. Danilo Ferreira E Silva

Nesta aula



- ☐ Instalar o Node.js.
- ☐ Instalar o pacote live-server.

| Próxima aula



- ☐ Instalar o VSCode.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 1.3. Instalação e configuração do VSCode

Prof. Danilo Ferreira E Silva

Nesta aula



- ☐ Instalar o VSCode.
- ☐ Instalar extensões:
 - ☐ Debugger for Chrome;
 - ☐ Prettier.
- ☐ Explorar o uso básico do VSCode.
- ☐ Editar configurações.

Próxima aula



- ☐ Criar e executar um primeiro projeto.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 1.4. Criando um primeiro projeto

Prof. Danilo Ferreira E Silva

Nesta aula



- ☐ Criar um projeto *Hello world*.
- ☐ Executar com o live-server.
- ☐ Executar no modo debug.

Conclusão



- ✓ Vimos uma visão geral do módulo.
- ✓ Instalamos as ferramentas necessárias.
- ✓ Criamos e executamos um primeiro projeto.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 2. HTML

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 2.1. HTML: Estrutura básica e sintaxe

Prof. Danilo Ferreira E Silva

Nesta aula



- ❑ O que é HTML.
- ❑ Estrutura de um documento.
- ❑ Sintaxe básica:
 - ❑ tags;
 - ❑ atributos;
 - ❑ texto.

HTML



HyperText Markup Language.

- ❑ Hipertexto: texto não linear, com diversos conteúdos interconectados (links).
- ❑ Define conteúdo e estrutura.
- ❑ Fontes de consulta:
 - ❑ MDN Web Docs (<https://developer.mozilla.org/>).
 - ❑ Especificação do W3C (<https://www.w3.org/html/>)

Estrutura de um documento



```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Meu documento</title>
  </head>
  <body>
    <p>Este é um documento HTML.</p>
  </body>
</html>
```

Estrutura de um documento

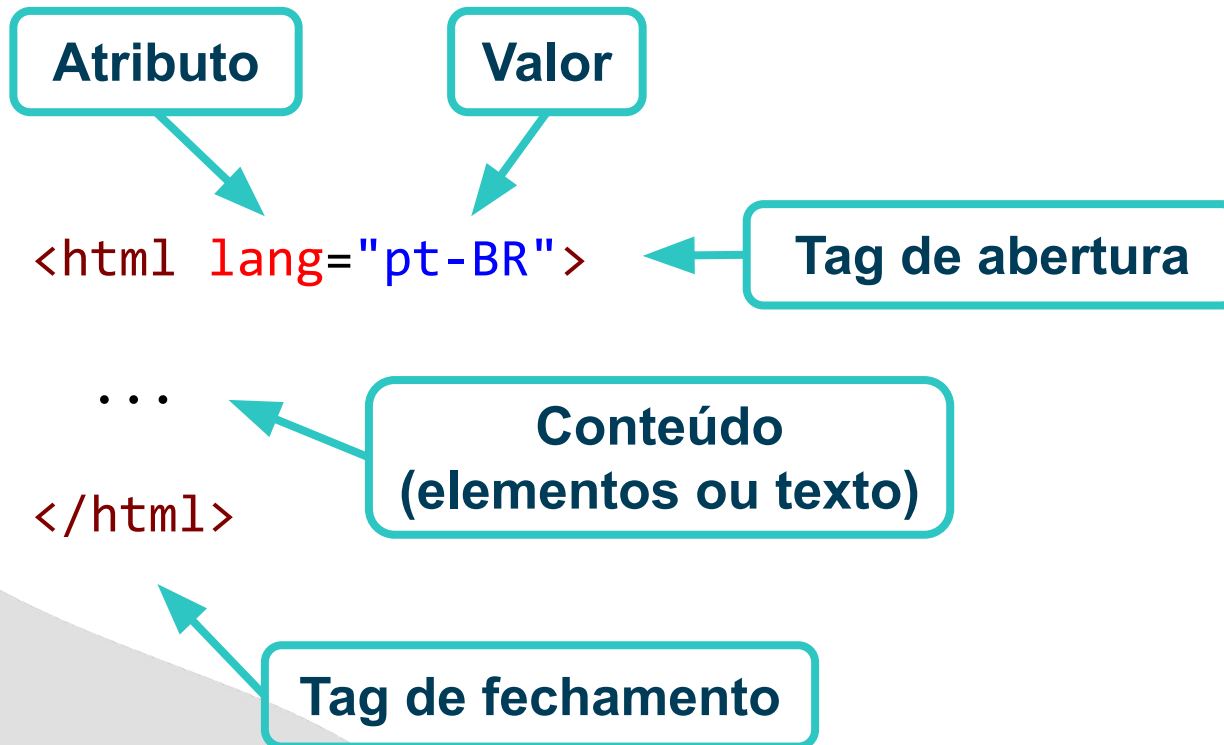


```
<!DOCTYPE html>  
<html lang="pt-BR">  
  <head>  
    <title>Meu documento</title>  
  </head>  
  <body>  
    <p>Este é um documento HTML.</p>  
  </body>  
</html>
```

Obrigatório

A purple rounded rectangle with the word 'Obrigatório' (Mandatory) in white text. Two purple arrows point from this box: one to the '<!DOCTYPE html>' declaration at the top of the code, and another to the '<title>' declaration within the head section of the code.

Sintaxe de um elemento



Escapar caracteres

IGTi

<	>
>	<
&	&

Elementos vazios



- ❑ Não possuem conteúdo, nem *tag* de fechamento.
- ❑ Podem opcionalmente terminar com `</>`

```
<input type="text">
```

```
<hr />
```

Metadados



```
<!DOCTYPE html>
<head lang="pt-BR">
  <meta charset="utf-8" />
  <title>Exemplo de HTML</title>
  <meta name="description" content="Descrição da página" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
</head>
<body>
  ...
</body>
```

Próxima aula



- ❑ Exemplos de elementos HTML mais usados.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 2.2. HTML: Texto, imagens e links

Prof. Danilo Ferreira E Silva

Nesta aula



- ☐ Ver exemplos de HTML com títulos, parágrafos, imagens, etc.
- ☐ Entender links relativos e absolutos.

| Próxima aula



- ❑ Mais exemplos de elementos HTML.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 2.3. HTML: Outros elementos

Prof. Danilo Ferreira E Silva

Nesta aula



- ☐ Ver exemplos de HTML com listas, tabelas e formulários.
- ☐ Explorar elementos para agrupamento de conteúdo (div, span, section, main, nav, footer, etc.).

Conclusão



- ✓ Entendemos a estrutura e sintaxe do HTML.
- ✓ Exploramos diversos elementos.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 3. CSS

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 3.1. Introdução a CSS e sintaxe básica

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ O que é CSS.
- ☐ Como adicionar CSS ao documento HTML.
- ☐ Sintaxe básica.

O que é CSS?



Cascading Style Sheets (folhas de estilo em cascata).

- ❑ **HTML**: define o conteúdo.
- ❑ **CSS**: define a estilização (cor, tamanho, posicionamento, etc.).

Adicionando CSS ao documento



1) Tag link

```
<!doctype html>  
<html>  
<head>  
  <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
  ...  
</body>  
</html>
```

Adicionando CSS ao documento



2) Tag style

```
<!doctype html>
<html>
<head>
  <style>
    div {
      color: blue;
    }
  </style>
</head>
<body>
  ...
</body>
</html>
```

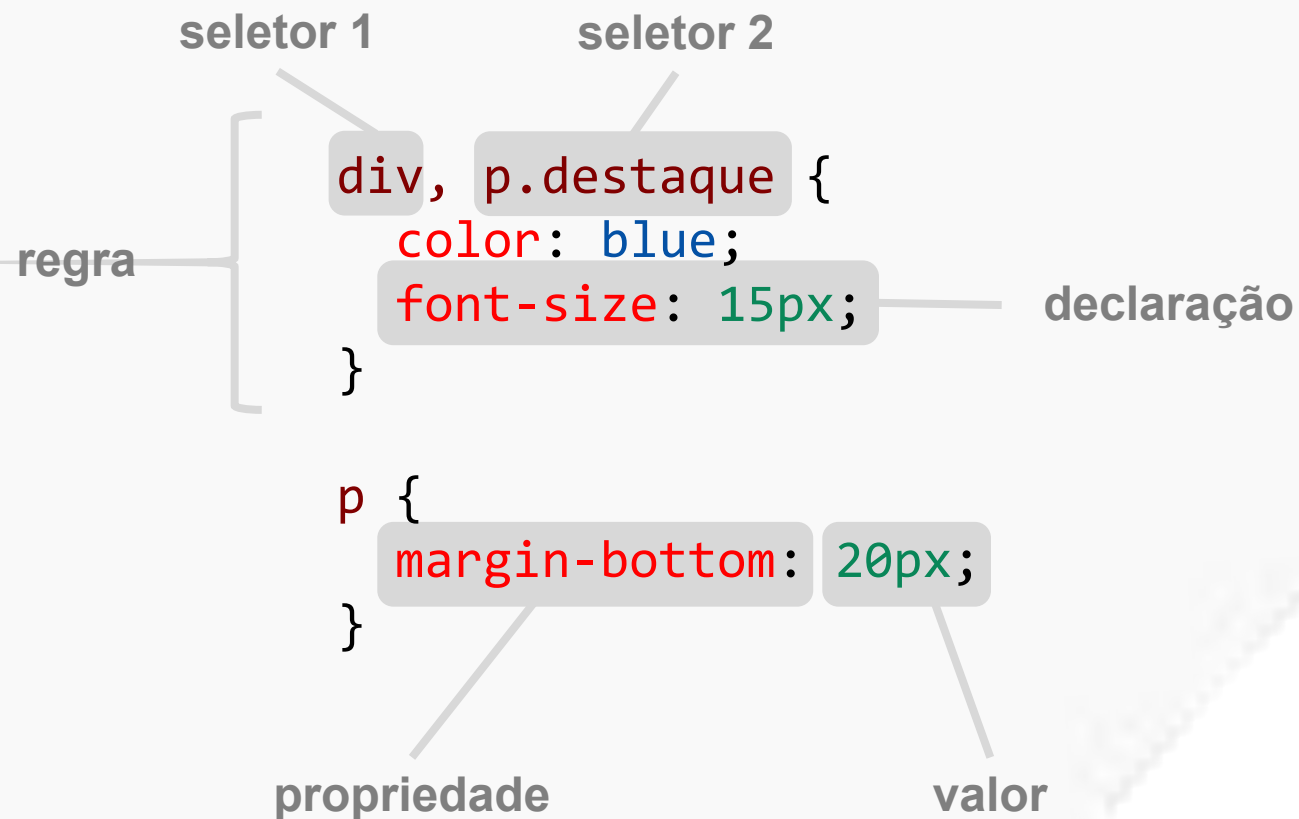
Adicionando CSS ao documento



3) Atributo style

```
<!doctype html>  
<html>  
<body>  
  <div style="color: blue"></div>  
</body>  
</html>
```


Sintaxe básica



Exemplos de propriedades



Herdadas	Não herdadas
color font-size font-family ...	margin width height border background-color ...

Estilos padrões do navegador



Título 1

Este é um parágrafo

Elements Console Sources Network >> ⚙️ ⋮ ✕

```
<!DOCTYPE html>
<html style>
  <head>...</head>
  <body style>
    ... <h1>Título 1</h1> == $0
    <p>Este é um parágrafo</p>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

html body h1

Styles Computed Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls + ⌵

element.style {

h1 {

display: block;

font-size: 2em;

margin-block-start: 0.67em;

margin-block-end: 0.67em;

margin-inline-start: 0px;

margin-inline-end: 0px;

font-weight: bold;

user agent stylesheet

⋮ Console ✕

CSS Reset



É comum utilizarmos um CSS pronto para normalizar os estilos padrões em diferentes navegadores.

❑ <https://meyerweb.com/eric/tools/css/reset/>

Próxima aula



- ☐ Seletores CSS.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 3.2. Seletores CSS

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Ver os diferentes tipos de seletores.
- ☐ Entender o mecanismo de especificidade de seletores.

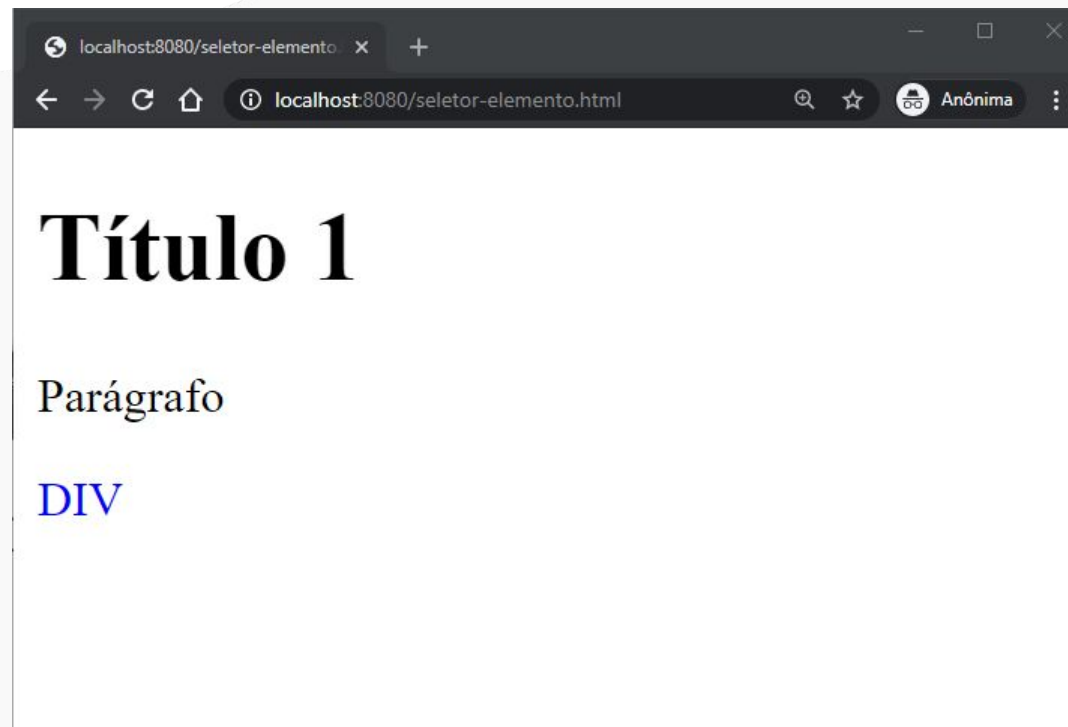
Seletor de elemento



E	Elemento do tipo E.
---	---------------------

```
<style>
  div {
    color: blue;
  }
</style>

<h1>Título 1</h1>
<p>Parágrafo</p>
<div>DIV</div>
```



Seletor de todos elementos



*	Todo elemento.
---	----------------

```
<style>
  * {
    color: blue;
  }
</style>

<h1>Título 1</h1>
<p>Parágrafo</p>
<div>DIV</div>
```



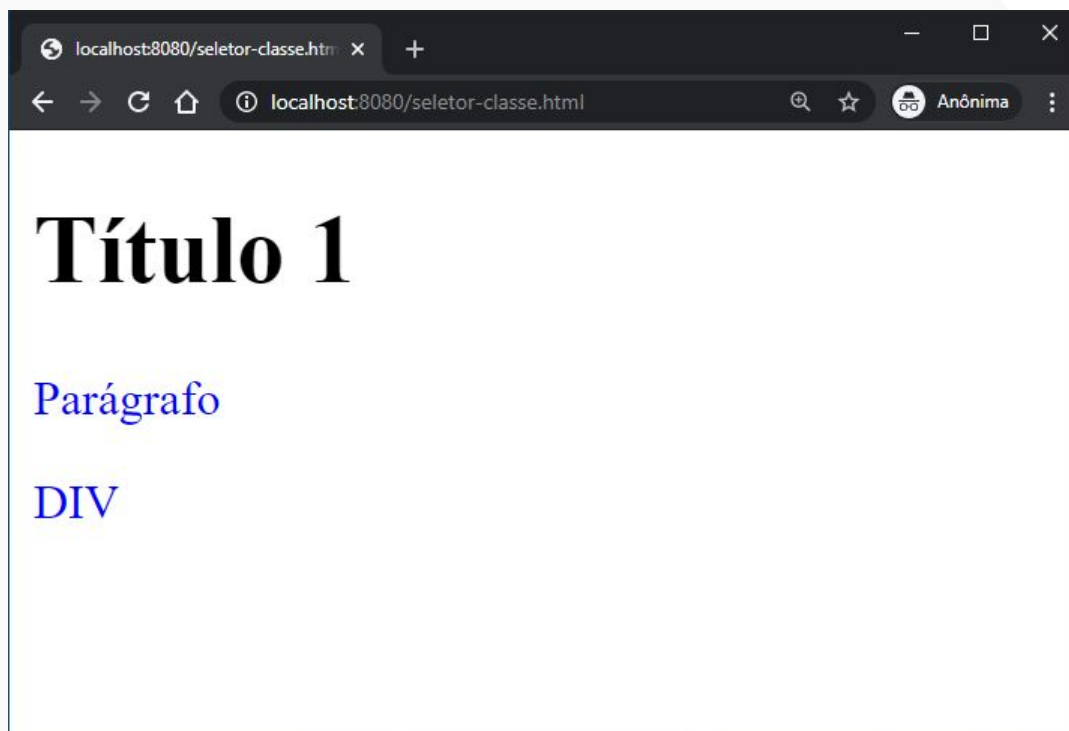
Seletor por classe



.classe	Elemento com a classe de nome classe .
----------------	---

```
<style>
  .azul {
    color: blue;
  }
</style>
```

```
<h1>Título 1</h1>
<p class="azul">Parágrafo</p>
<div class="azul">DIV</div>
```



Seletor por ID



#id	Elemento com atributo <i>id</i> igual a id .
-----	---

```
<style>
  #t1 {
    color: blue;
  }
</style>

<h1 id="t1">Título 1</h1>
<p>Parágrafo</p>
<div>DIV</div>
```

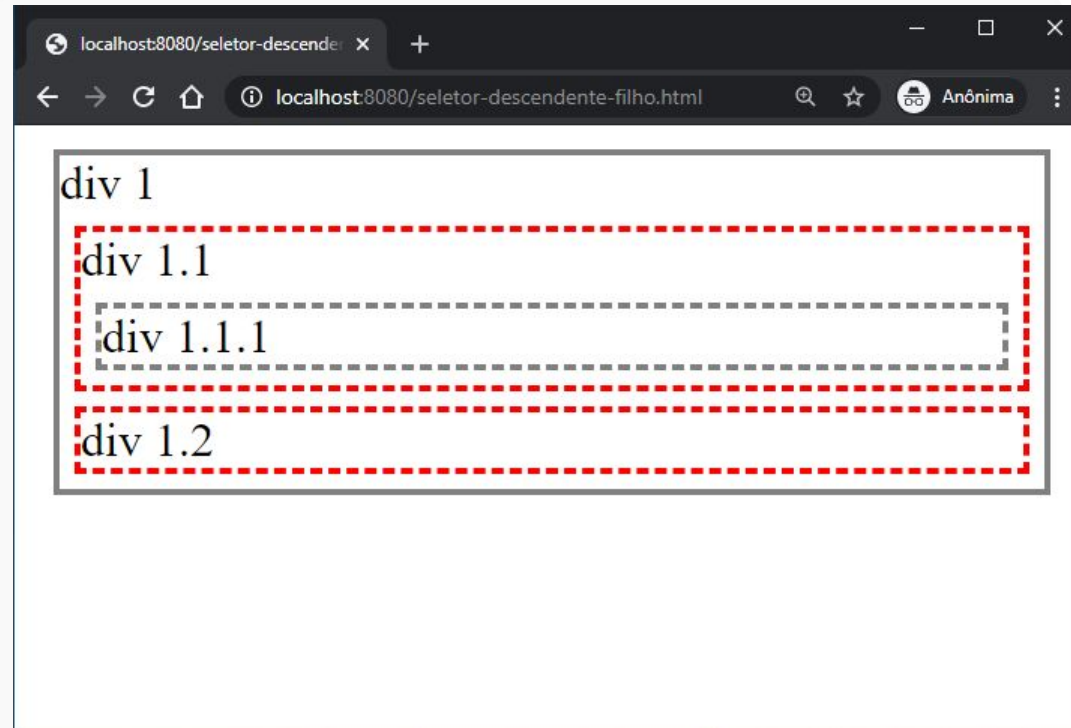


Combinador descendente e filho



```
<style>
  div {
    border: 2px solid grey;
    margin: 5px;
  }
  .c1 div {
    border-style: dashed;
  }
  .c1 > div {
    border-color: red;
  }
</style>
<div class="c1">
  div 1
  <div>
    div 1.1
    <div>div 1.1.1</div>
  </div>
  <div>div 1.2</div>
</div>
```

E F	Elemento F descendente de E.
E > F	Elemento F filho de E.

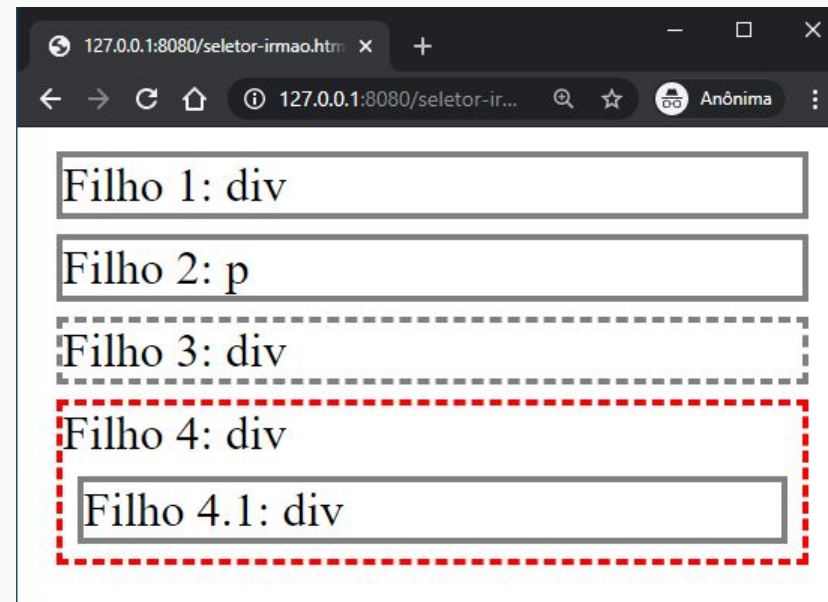


Combinador + e ~



```
<style>
  div, p {
    border: 2px solid grey;
    margin: 5px;
  }
  div ~ div {
    border-style: dashed;
  }
  div + div {
    border-color: red;
  }
</style>
<body>
  <div>Filho 1: div</div>
  <p>Filho 2: p</p>
  <div>Filho 3: div</div>
  <div>Filho 4: div
    <div>Filho 4.1: div</div>
  </div>
</body>
```

E ~ F	Elemento F é irmão de E e precedido por E .
E + F	Elemento F irmão de E e <i>imediatamente</i> precedido por E .



Pseudoclasses



:disabled	Elemento de interface que esteja desabilitado.
:hover	Elemento cujo mouse está sobre.
:focus	Elemento que possui foco do cursor do teclado.
:first-child	Elemento que é o primeiro filho de seu pai.
:last-child	Elemento que é o último filho de seu pai.
:empty	Elemento que não possui filho.
:not(E)	Elemento <i>não</i> corresponde ao seletor E

Veja mais em <https://www.w3.org/TR/selectors-4/>

Seletores compostos



```
a.destaque.grande:disabled
```

Elemento **A**, com as classes **destaque** e **grande**, que está desabilitado.

Seletores complexos



`table.comBordas tr + tr`

Elemento **TR**, imediatamente precedido por um irmão que é um elemento **TR** descendente de um elemento **TABLE** com a classe **comBordas**.

Especificidade de seletores



O que ocorre quando seletores de regras distintas capturam o mesmo elemento?

Especificidade de seletores



```
button {  
  border: 2px solid black;  
  background-color: #909090;  
}  
.destaque {  
  color: white;  
  background-color: #1ba3da;  
}
```

Propriedades aplicadas

```
border: 2px solid black;  
color: white;  
background-color: ???;
```

```
<button class="destaque">Teste</button>
```

Especificidade de seletores



Critérios para definir a prioridade:

1. Declarações dentro do atributo **style**.
2. Quantidade de seletores por ID da regra.
3. Quantidade de seletores por classe ou pseudoclasse da regra.
4. Quantidade de seletores por elemento da regra.
5. Ordem de declaração da regra (última é aplicada)

Especificidade de seletores



```
button {  
  border: 2px solid black;  
  background-color: #909090;  
}  
.destaque {  
  color: white;  
  background-color: #1ba3da;  
}
```

Propriedades aplicadas

```
border: 2px solid black;  
color: white;  
background-color: ???;
```

```
<button class="destaque">Teste</button>
```

Especificidade de seletores



```
button {  
  border: 2px solid black;  
  background-color: #909090;  
}  
.destaque {  
  color: white;  
  background-color: #1ba3da;  
}
```

Propriedades aplicadas

```
border: 2px solid black;  
color: white;  
background-color: #1ba3da;
```

```
<button class="destaque">Teste</button>
```

| Próxima aula



- ❑ Exercício de seletores CSS.

Fundamentos

Módulo 2. Bootcamp Desenvolvedor Front End

Aula 3.2.1. Desafio guiado: usar seletores para estilizar uma lista

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Desafio guiado: usar seletores para estilizar uma lista (elementos UL e LI).

Próxima aula



- ❑ Tamanho e posicionamento de elementos.

Fundamentos

Módulo 2. Bootcamp Desenvolvedor Front End

Aula 3.3. Dimensionamento e posicionamento de elementos

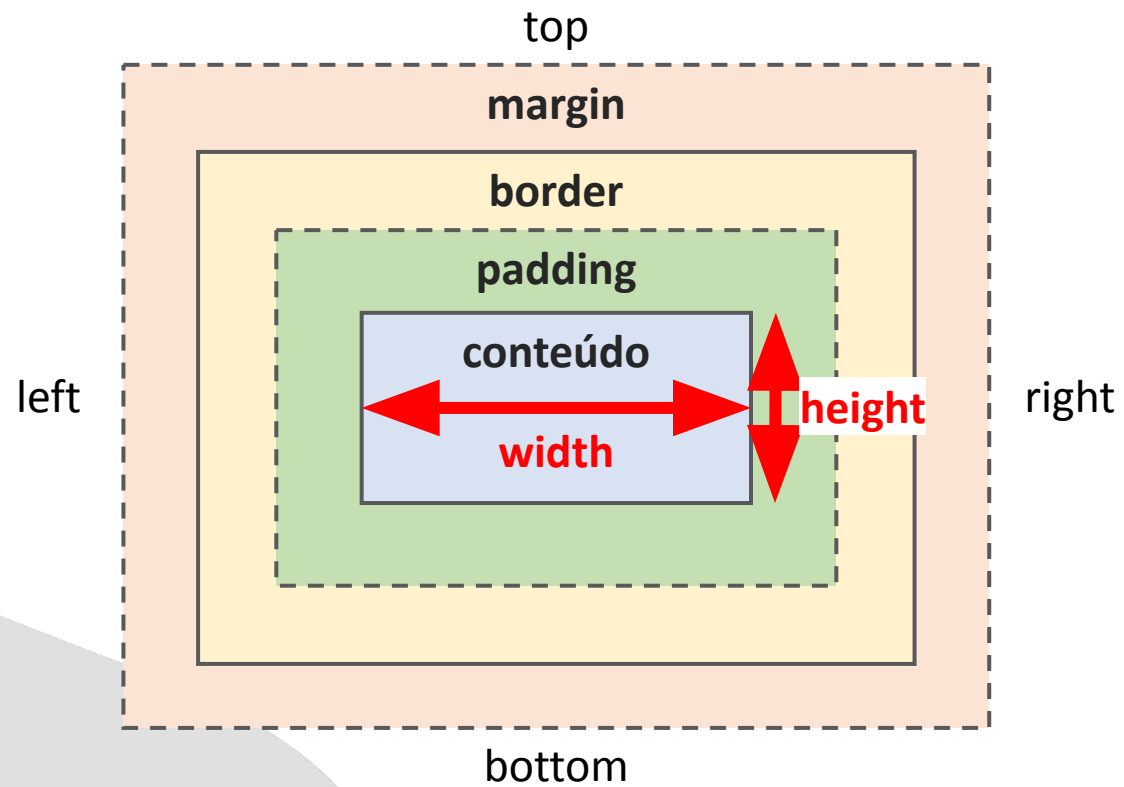
Prof. Danilo Ferreira e Silva

Nesta aula



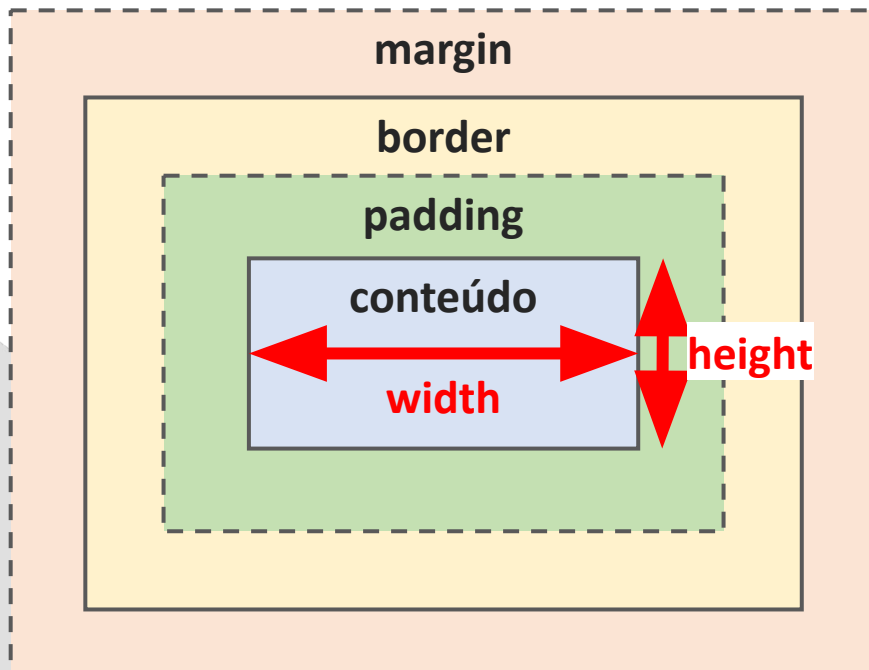
- ❑ Estudar dimensionamento de elementos:
 - ▮ *box model*.
 - ▮ Unidades de medida mais usadas.
- ❑ Entender como os elementos são posicionados na página:
 - ▮ *Normal flow*.
 - ▮ *text-align*.
 - ▮ *Baseline* e *vertical-align*.

Box model



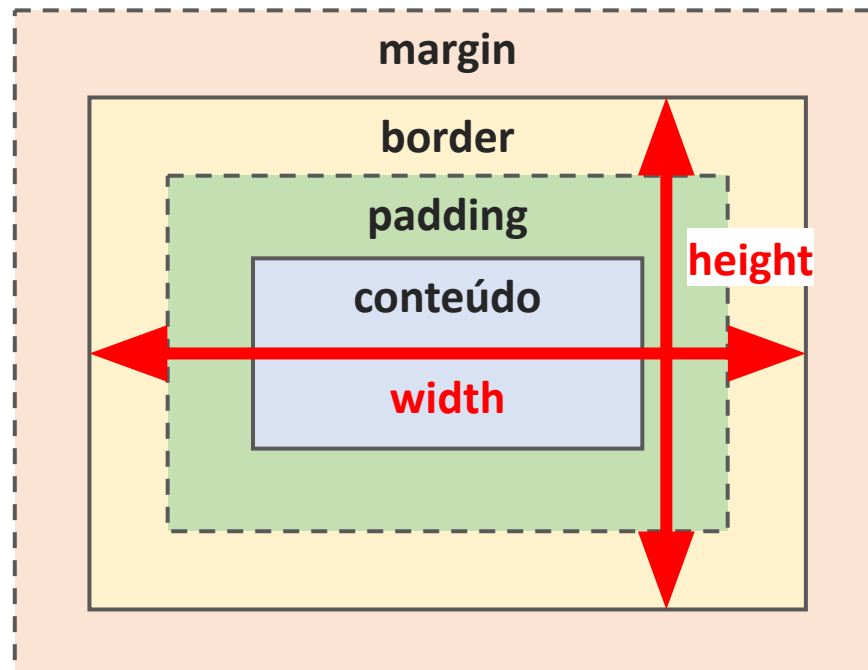
content-box vs. border-box

```
div {  
  box-sizing: content-box;  
}
```



Padrão

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



Algumas unidades de medida



px	Um ponto da tela
cm, mm, in	Centímetros, milímetros, polegadas, etc. (fazem mais sentido para impressão)
em	Tamanho da fonte corrente.
rem	Tamanho da fonte do elemento raiz da página.
vh	1% da altura do <i>viewport</i> .
vw	1% da largura do <i>viewport</i> .
%	Relativo a outra medida (normalmente do elemento pai)

Propriedade overflow



- ❑ **visible** (padrão): conteúdo pode ser renderizado fora do *box*.
- ❑ **hidden**: conteúdo é cortado.
- ❑ **scroll**: exibe scroll bar sempre
- ❑ **auto**: exibe scroll bar, se precisar

Posicionamento de *boxes*



- ❑ Se não especificado, segue o *normal flow* (*flow layout*).
- ❑ Dois tipos de *boxes*:
 - ▢ *block box*;
 - ▢ *inline box*.

Block box



- ❑ Exibido em uma nova linha.
- ❑ Largura é 100% do espaço disponível em seu contêiner, por padrão.
- ❑ Exemplos: **h1**, **div**, **p**, **ul**, **li**, etc.

Inline box



- ❑ Exibido na mesma linha (pode ter quebra de linha se não houver espaço).
- ❑ **width** e **height** não se aplicam.
- ❑ **margin**, **padding** e **border** verticais se comportam diferente (não “empurram” os boxes vizinhos).
- ❑ Exemplos: **span**, **a**, **img**, **em**, **strong**, etc.

Propriedade text-align



- ❑ Controla o alinhamento horizontal de *inline boxes* (não funciona para **block boxes**).
- ❑ Valores comuns: **left**, **right**, **center**, **justify**.

Propriedade vertical-align



- ❑ Controla o alinhamento vertical de *inline boxes*, ou células de tabelas.
- ❑ Valores comuns: **baseline**, **middle**, **top**, **bottom**.
- ❑ Não são intuitivos como parecem, é necessário entender o conceito de *baseline*.

vertical-align: baseline (padrão)



xpto xpto xpto xpto xpto xpto



baseline

xpto grande xpto pequeno

xpto xpto xpto xpto xpto xpto xpto

vertical-align: middle | top | bottom

igti

middle  ~~xpto~~

The diagram shows a yellow square divided into four quadrants by a blue cross. A red horizontal line passes through the center of the square, aligning with the middle of the text 'xpto'.

top  xpto

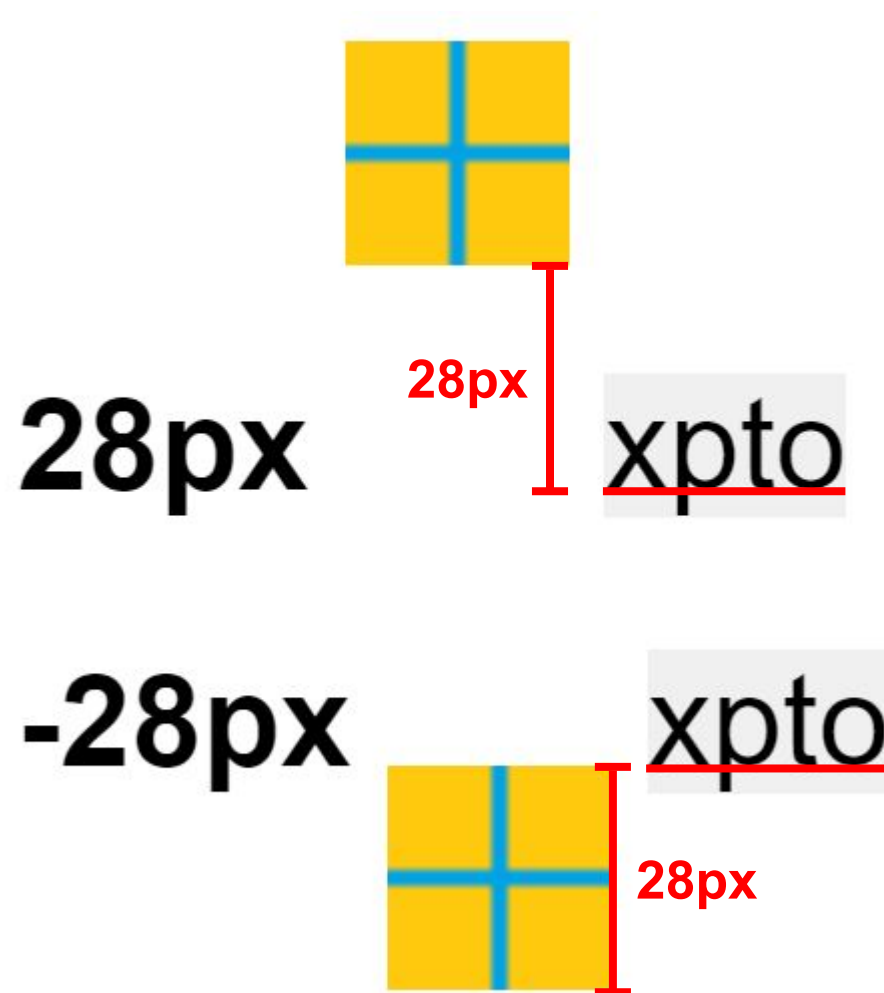
The diagram shows a yellow square divided into four quadrants by a blue cross. A red horizontal line is at the top edge of the square, aligning with the top of the text 'xpto'.

bottom  xpto

The diagram shows a yellow square divided into four quadrants by a blue cross. A red horizontal line is at the bottom edge of the square, aligning with the bottom of the text 'xpto'.

vertical-align: <dimensão>

IGTi



Propriedade display



- ❑ É possível definir, via propriedade **display**, se um elemento é um *block box* ou *inline box*.

```
div {  
  display: inline;  
}
```

```
span {  
  display: block;  
}
```


Block dentro de inline



- ❑ É incorreto colocar um elemento com display **block** dentro de um elemento com display **inline** (o *inline box* é quebrado).
- ❑ Para isso existe o **display: inline-block**
 - ❑ Se comporta como *inline* externamente
 - ❑ Se comporta como *block* internamente (aceita width, height, etc.)

| Próxima aula



- ❑ Posicionamento fora do normal flow.

Fundamentos

Módulo 2. Bootcamp Desenvolvedor Front End

Aula 3.4. Posicionamento fora do normal flow

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Como posicionar elementos fora do *normal flow*.

Posicionamento fora do *normal flow*



- ❑ Controlado pela propriedade **position**.
- ❑ Valor **static** é o padrão (*normal flow*).
- ❑ Fora do *normal flow*:
 - ❑ **relative**
 - ❑ **absolute**
 - ❑ **fixed**
 - ❑ **sticky**

Position: relative



- ❑ Posicionado de acordo com *normal flow*, mas deslocado de sua posição normal via propriedades **top**, **right**, **bottom** e **left**.

Position: absolute



- ❑ Posicionado fora do *normal flow* (não ocupa espaço).
- ❑ Posição definida via propriedades **top**, **right**, **bottom** e **left**, relativas ao primeiro elemento pai **posicionado** (position \neq static).
- ❑ *Block boxes* não se expandem para largura disponível no pai.

Position: fixed



- ❑ Posicionado fora do *normal flow* (não ocupa espaço).
- ❑ Posição definida via propriedades **top**, **right**, **bottom** e **left**, relativas ao *viewport* (não faz scroll).
- ❑ *Block boxes* não se expandem para largura disponível no pai.

| Próxima aula



- ❑ Flexbox layout.

Fundamentos

Módulo 2. Bootcamp Desenvolvedor Front End

Aula 3.5. Flexbox layout

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Conhecer o modelo de layout **flexbox**.
 - ❑ Introduzido nas especificações mais recentes do CSS.
 - ❑ Muito versátil e poderoso.

display: flex | inline-flex

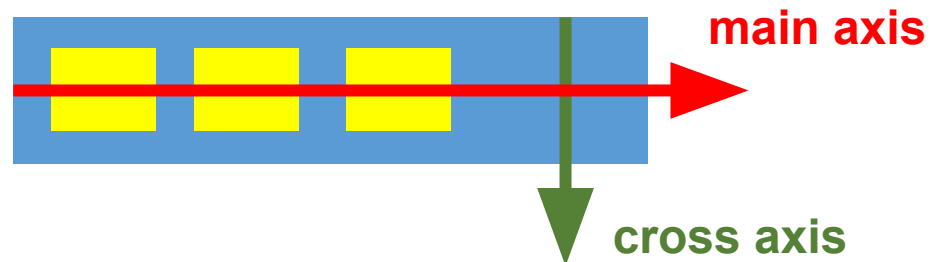


- ❑ Habilitado por meio da propriedade display:
 - ❑ **flex**: comporta como *block box* externamente, e *flexbox layout* internamente;
 - ❑ **inline-flex**: comporta como *inline box* externamente, e *flexbox layout* internamente.

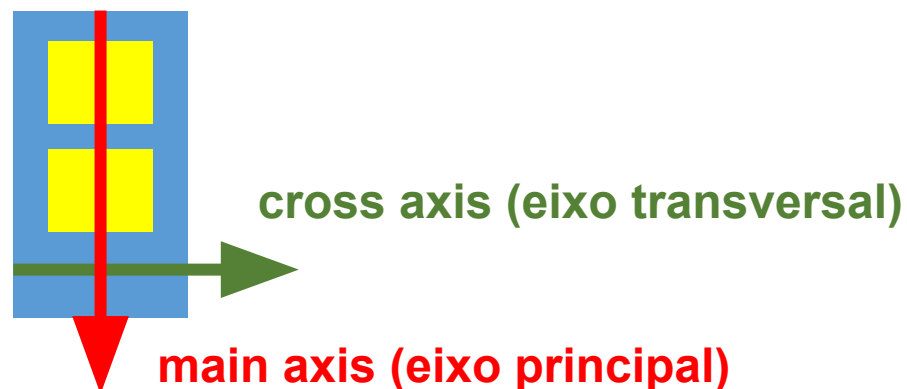
Flexbox layout

- ❑ Elementos filhos exibidos em linhas ou colunas, de acordo com a propriedade **flex-direction**:

- ❑ **row** (padrão);



- ❑ **column.**



Flexbox layout



- ❑ É possível controlar:
 - ❑ Alinhamento no eixo principal (**justify-content**);
 - ❑ Alinhamento no eixo transversal (**align-items**, **align-self**);
 - ❑ Tamanho de cada item (**flex**, **flex-grow**, **flex-shrink**, **flex-basis**);
 - ❑ Quebras de linha/colunas (**flex-wrap**).

Próxima aula



- ❑ Imitar o layout do site *Stack Overflow*.

Fundamentos

Módulo 2. Bootcamp Desenvolvedor Front End

Aula 3.5.1. Desafio guiado: imitar o layout do site Stack Overflow

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Desafio guiado: imitar o layout do site *Stack Overflow*.

Fundamentos

Módulo 2. Bootcamp Desenvolvedor Front End

Aula 3.5.2. Desafio guiado: imitar o layout do site Stack Overflow (parte 2)

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Desafio guiado: imitar o layout do site *Stack Overflow* (parte 2).

Fundamentos

Módulo 2. Bootcamp Desenvolvedor Front End

Aula 3.5.3. Desafio guiado: imitar o layout do site Stack Overflow (parte 3)

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Desafio guiado: imitar o layout do site *Stack Overflow* (parte 3).

Conclusão



- ✓ Sintaxe básica, propriedades, herança.
- ✓ Seletores.
- ✓ Dimensionamento.
- ✓ Posicionamento no *normal flow*.
- ✓ Posicionamento fora do *normal flow*.
- ✓ Flexbox layout.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 4. JavaScript básico

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 4.1. Introdução a JavaScript

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Principais características da linguagem JavaScript.
- ❑ Como adicionar scripts ao documento HTML.
- ❑ Sintaxe básica:
 - ❑ declaração de variáveis e atribuição de valores;
 - ❑ chamadas de funções;
 - ❑ comentários.

JavaScript



- ❑ Linguagem de programação dinâmica e interpretada.
- ❑ Executa no navegador (e outras plataformas).
- ❑ Surgiu com o navegador Netscape.
- ❑ Segue a especificação ECMAScript
(<https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>).

Histórico



- ❑ ...
- ❑ ECMAScript 5.1 (ES5).
- ❑ **ECMAScript 6 (ES6, ou ES2015).**
- ❑ ECMAScript 2016 (ES2016).
- ❑ ...
- ❑ ECMAScript 2020 (ES2020).

Adicionar JavaScript na página



- ☐ Script externo.

```
<script src="./js/meu-script.js"></script>
```

- ☐ Script *Inline*.

```
<script>  
  console.log('Este é um script');  
</script>
```

Próxima aula



- ❑ Tipos e valores em JavaScript.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 4.2. Tipos e valores em JavaScript

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Tipos primitivos (number, boolean, string).
- ❑ Objetos.
- ❑ Arrays.
- ❑ Os valores *null* e *undefined*.
- ❑ Cópia por valor vs. referência.

| Próxima aula



- ☐ Operadores e expressões.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 4.3. Operadores e expressões

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Operadores:
 - ❑ lógicos;
 - ❑ aritméticos;
 - ❑ comparação.
- ❑ Precedência de operadores e parênteses.

Operadores lógicos



!	Negação
&&	E lógico
	OU lógico

Operadores aritméticos



$x + y$	Adição
$x - y$	Subtração
$+x$	Converte para número (se já não for)
$-x$	Inversão de sinal
x / y	Divisão
$x * y$	Multiplicação
$x \% y$	Resto da divisão de x por y
$x ** y$	Exponenciação

Operadores aritméticos com atribuição



<code>x += y</code>	Adição
<code>x -= y</code>	Subtração
<code>x /= y</code>	Divisão
<code>x *= y</code>	Multiplicação
<code>x %= y</code>	Resto da divisão de x por y
<code>x **= y</code>	Exponenciação

Operadores de incremento/decremento



<code>x++</code>	Retorna x e depois incrementa
<code>x--</code>	Retorna x e depois decrementa
<code>++x</code>	Incrementa e depois retorna x
<code>--x</code>	Decrementa e depois retorna x

Comparação



$x < y$	Menor
$x > y$	Maior
$x \leq y$	Menor ou igual
$x \geq y$	Maior ou igual

Igualdade



<code>x == y</code>	Igual
<code>x != y</code>	Diferente
<code>x === y</code>	Exatamente igual
<code>x !== y</code>	Não exatamente igual

Precedência de operadores



Precedência

Operadores	Associatividade
$f(x)$ $x.y$ $x[y]$	<input type="checkbox"/>
$!x$ $+x$ $-x$	<input type="checkbox"/>
$x**y$	<input type="checkbox"/>
$x * y$ x / y $x \% y$	<input type="checkbox"/>
$x + y$ $x - y$	<input type="checkbox"/>
$x < y$ $x <= y$ $x > y$ $x >= y$	<input type="checkbox"/>
$x == y$ $x != y$ $x === y$ $x !== y$	<input type="checkbox"/>
$x \&\& y$	<input type="checkbox"/>
$x y$	<input type="checkbox"/>

| Próxima aula



- ☐ Funções e escopo.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 4.4. Funções e escopo

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Declaração de funções.
- ☐ Parâmetros e valor de retorno.
- ☐ Variáveis locais e globais.

| Próxima aula



- ❑ Comandos de decisão.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 4.5. Comandos de decisão

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Comando if.
- ☐ Operador ternário.
- ☐ Comandos switch e case.

| Próxima aula



- ❑ Comandos de repetição.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 4.6. Comandos de repetição

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Comandos `while` e `do while`.
- ❑ Comando `for`.

Próxima aula



- ❑ Implementar uma função usando os recursos já estudados.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 4.6.1. Desafio guiado: função palíndromo

Prof. Danilo Ferreira e Silva

Conclusão



Aprendemos o básico para programação em JavaScript.

- ✓ Sintaxe básica.
- ✓ Variáveis e valores.
- ✓ Funções.
- ✓ Comandos de repetição e decisão.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 5. Interação com o DOM

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 5.1. Interação básica com o DOM

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ O que é o DOM.
- ❑ Como modificar elementos na página e reagir a eventos via JavaScript.

DOM



- ❑ Document **O**bject **M**odel.
- ❑ Estrutura de dados em forma de árvore que representa o documento.
- ❑ Podemos interagir com os nós da árvore (elementos) via JavaScript:
 - ❑ Ler e alterar propriedades;
 - ❑ Registrar *listeners* de eventos;
 - ❑ Inserir/excluir.

Próxima aula

- ☐ Falar mais sobre eventos.

IGTi

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 5.2. Eventos

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Entender a função `addEventListener`.
- ☐ Entender a propagação de eventos.
- ☐ Entender funções `stopPropagation` e `preventDefault`.

Tratando eventos



1. No HTML, via atributos.

```
<button id="mybtn" onclick="myHandler()">Click me</div>
```

2. Em JavaScript, via addEventListener.

```
var el = document.getElementById("mybtn");  
el.addEventListener("click", myHandler);
```

removeEventListener



- ❑ É possível remover um *handler* de eventos, se for necessário.

```
var el = document.getElementById("mybtn");  
el.addEventListener("click", myHandler);
```


...

```
el.removeEventListener("click", myHandler);
```

Propagação de eventos



```
<body>
  <div id="d1">
    <div id="d2"></div>
  </div>
</body>
```



1

Usuário clica em #d2.

2

Fase de **capturing**.

Executa click handlers de:

1. body;
2. #d1;
3. #d2.

3

Fase de **bubbling**.

Executa click handlers de:

1. #d2;
2. #d1;
3. body.

Propagação de eventos



```
// registra na fase de capturing  
el.addEventListener("click", myHandler, true);  
  
// registra na fase de bubbling  
el.addEventListener("click", myHandler, false);
```


Objeto evento



```
function myHandler(event) {  
  console.log(  
    "cliqueu na posição " +  
    event.pageX + ", " + event.pageY  
  );  
}
```

A screenshot of the MDN Web Docs page for the `MouseEvent` interface. The page header includes the MDN logo, navigation links for Technologies, References & Guides, and Feedback, a search bar, and a sign-in link. The main heading is `MouseEvent`. Below it, a breadcrumb trail shows the path: Web technology for developers > Web APIs > MouseEvent. A language selector is set to English (US). The page is divided into several sections: **Table of contents** (with links to Constructor, Properties, Constants, Methods, Example, Specifications, Browser compatibility, and See also), **Related Topics** (with a link to `MouseEvent`), **Constructor** (describing the `MouseEvent()` method), and **Properties** (noting that the interface inherits from `UIEvent` and `Event`). A diagram illustrates the inheritance hierarchy: `Event` is the base, which `UIEvent` inherits from, and `MouseEvent` inherits from `UIEvent`. The `MouseEvent` section also includes a brief description of the interface and a list of common events like `click`, `dblclick`, `mouseup`, and `mousedown`.

`event.stopPropagation`



- ❑ Se um *handler* de evento chama a função `event.stopPropagation`, a propagação do evento é interrompida no ponto que está.

event.preventDefault



- ❑ Se um *handler* de evento chama a função `event.preventDefault`, o navegador não executa a ação padrão associada àquele evento, por ex.:
 - ❑ Abrir um link no evento **click**;
 - ❑ Submeter um formulário no evento **submit**.

Alguns eventos comuns



focus	Elemento recebe foco.
blur	Elemento perde foco.
input	Valor de um elemento muda (input, select, textarea).
submit	Formulário submetido.
keydown	Tecla do teclado pressionada.
keyup	Tecla do teclado liberada.
click	Botão do mouse pressionado e liberado.
mousemove	Mouse movimentado sobre o elemento.

Próxima aula



- ☐ Criar elementos dinamicamente.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 5.3. Criando elementos dinamicamente

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Entender a função `document.createElement`.
- ☐ Entender funções `appendChild`, `remove` e `insertBefore`.

Próxima aula



- ☐ Alterar estilos de elementos.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 5.4. Alterando estilos de elementos

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Alterar classes e estilos de elementos dinamicamente, usando as propriedades:

- ❑ `element.style;`

- ❑ `element.className;`

- ❑ `element.classList.`

Conclusão



Aprendemos a manipular o DOM.

- ✓ Obter referências para elementos.
- ✓ Alterar propriedades.
- ✓ Tratar eventos.
- ✓ Adicionar/remover elementos.
- ✓ Alterar estilos.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 6. Orientação a objetos em JavaScript

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 6.1. Instanciando objetos

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Instanciar objetos com o operador **new**.
- ☐ Entender a referência **this**.

Classes e objetos



- ❑ Em linguagens orientada a objetos, normalmente definimos **classes**, e usamos seu **construtor** para instanciar objetos.
- ❑ Em JavaScript, qualquer função pode ser usada para instanciar um objeto por meio do operador **new**.

Próxima aula



- ☐ Entender o mecanismo *Prototype chain*.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 6.2. Prototype chain

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Entender a *prototype chain*, responsável pela herança de propriedades em objetos.

Herança de propriedades



- ❑ Objetos possuem propriedades próprias, mas também podem herdar propriedades de outro objeto, o seu *prototype*.

Prototype chain



- ❑ Ao acessar uma propriedade:
 - ❑ busca-se no objeto atual;
 - ❑ depois em seu *prototype*;
 - ❑ depois no *prototype* do *prototype*;
 - ❑ ...
 - ❑ até que não exista um *prototype*.

Alterando o prototype de um construtor



```
function MyConstructor() {  
    // ...  
}  
MyConstructor.prototype.myProp = 'some value';  
  
var myObj = new MyConstructor();  
console.log(myObj.myProp); // imprime 'some value'
```

| Próxima aula



- ❑ Classes e herança.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 6.3. Classes e herança

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Conhecer a sintaxe moderna para declarar classes, introduzida do ES6.
- ☐ Utilizar herança entre classes.

Conclusão



Aprendemos recursos de JavaScript que possibilitam programação orientada a objetos.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 7. JavaScript moderno

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 7.1. Let, const, desestruturação, spread e template strings

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Conhecer recursos modernos da linguagem, introduzidos no ES6:
 - ❑ Declaração de variáveis com `let` e `const`;
 - ❑ Atribuição via desestruturação;
 - ❑ Spread operator;
 - ❑ Template strings (ou template literals).

| Próxima aula



- ❑ Arrow functions.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 7.2. Arrow functions

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Conhecer a sintaxe de declaração de funções mais compacta, conhecida como *arrow functions*, introduzida no ES6.

| Próxima aula



- ❑ Manipulação de arrays.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 7.3. Manipulação de arrays

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Estrutura de repetição for ... of.
- ❑ Conhecer funções de manipulação de arrays, como:
 - ❑ `forEach;`
 - ❑ `map;`
 - ❑ `filter;`
 - ❑ `find;`
 - ❑ `sort.`

Próxima aula



- ☐ Módulos.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 7.4. Módulos

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Conhecer o mecanismo de módulos introduzido no ES6.

Por que módulos?



- ❑ Quando a aplicação fica muito grande, torna-se necessário modularizar.
- ❑ Separar o código em vários scripts traz alguns problemas.
 - ❑ Dependências entre scripts são implícitas.
 - ❑ Não há encapsulamento adequado (tudo que é declarado no escopo global pode ser lido/escrito de qualquer script).

Módulos (ESM)



modulo_a.js

```
function fazAlgo() {  
  // ...  
}  
  
export { fazAlgo };
```

modulo_b.js

```
import { fazAlgo } from "../modulo_a.js";  
  
fazAlgo();
```

Conclusão



Aprendemos diversos recursos de ES6.

- ✓ `let`, `const` e desestruturação.
- ✓ *Rest operator*.
- ✓ *Template literals*.
- ✓ *Arrow functions*.
- ✓ `for ... of` e funções de arrays.
- ✓ Módulos.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 8. Requisições HTTP em JavaScript

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 8.1. A API fetch

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Aprender a fazer requisições HTTP via JavaScript com a API fetch.

Por que fazer requisições HTTP?



- ❑ Carregar dados dinamicamente (normalmente JSON), sem a necessidade de carregar outra página completa.
 - ❑ É possível implementar a aplicação inteira com um único documento HTML, carregando dados dinamicamente via JavaScript.

fetch



```
fetch("http://minha.api")
```

- ☐ Executa de forma assíncrona.
- ☐ Resposta é encapsulada em uma Promise.

Opções do fetch



```
fetch("http://minha.api", { /* opções */ })
```

- ❑ **method**: GET, POST, PUT, DELETE, HEAD, etc.
- ❑ **headers**: cabeçalhos HTTP.
- ❑ **body**: corpo da requisição (postar dados codificados em json, por exemplo).
- ❑ ...

Próxima aula



- ☐ Aprender mais sobre *promises*.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 8.2. Dominando promises: carregamento sequencial e paralelo

Prof. Danilo Ferreira e Silva

Nesta aula



- ☐ Carregamento sequencial de dados.
- ☐ Simplificação do código usando encadeamento de *promises*.
- ☐ Carregamento paralelo de dados usando `Promise.all`.

| Próxima aula



▣ *Async/await.*

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 8.3. Async/await

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Aprender a usar o recurso *async/await*, introduzido no ES2017, para simplificar código com *promises*.

| Próxima aula



- ☐ Tratamento de erros.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 8.4. Tratamento de erros

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Aprender a tratar erros no carregamento de dados:
 - ❑ usando a função `promise.catch`;
 - ❑ usando *try/catch*.
- ❑ Aprender a usar `finally`.

Próxima aula



- ❑ Desafio guiado: Implementar CRUD completo.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 8.4.1. Desafio guiado: implementar CRUD completo

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Implementar uma aplicação completa de cadastro de funcionários, no estilo CRUD (create, retrieve, update, delete), usando o Back End fornecido.

Operações



```
// Criar
fetch(`http://localhost:3000/employees`, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify(employee),
});
```

```
// Atualizar
fetch(`http://localhost:3000/employees/${id}`, {
  method: "PUT",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify(employee),
});
```

```
// Excluir
fetch(`http://localhost:3000/employees/${id}`, {
  method: "DELETE",
});
```

Conclusão



- ✓ Carregamos dados usando *fetch*.
- ✓ Trabalhamos com *promises*.
- ✓ Aprendemos o uso de *async/await* para simplificar o código.
- ✓ Aprendemos a tratar erros.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Capítulo 9. Tarefas temporizadas ou periódicas em JavaScript

Prof. Danilo Ferreira e Silva

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 9.1. `setTimeout`

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Aprender a agendar a execução de uma tarefa após um intervalo de tempo, usando a função `setTimeout`.

Próxima aula



- ❑ Ver a função `setInterval`.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 9.2. setInterval

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Aprender a executar uma tarefa repetidamente, usando a função `setInterval`.

Próxima aula



- ❑ Executar animações com `requestAnimationFrame`.

Fundamentos

Módulo 1. Bootcamp Desenvolvedor Front End

Aula 9.3. `requestAnimationFrame`

Prof. Danilo Ferreira e Silva

Nesta aula



- ❑ Aprender a executar animações usando a função `requestAnimationFrame`.

Conclusão



- ✓ Execução de tarefa com temporizador.
- ✓ Execução de tarefa periódica.
- ✓ Execução de tarefa antes da renderização de cada quadro, de acordo com a taxa de atualização da tela.