



## DEFINIÇÃO

Conceito de Folhas de estilo/CSS. Recursos disponíveis nas Folhas de estilo/CSS. Conceito de Box Model em uma página web. Aplicação de CSS a Boxes em uma página web. Conceito de seletores CSS. Conceito de pseudoclasses e pseudo-elementos. Aplicação de estilos CSS utilizando pseudoclasses e pseudo-elementos. Propriedades CSS de posicionamento. Conceito de frameworks CSS. Apresentação de frameworks CSS.

## PROPÓSITO

Compreender o que são Folhas de estilo – CSS e como podem ser utilizadas para cuidar da apresentação, layout e estilo de páginas HTML.

## PREPARAÇÃO

Para aplicação dos exemplos, será necessário um editor de texto com suporte à marcação HTML. No sistema operacional Windows, é indicado o Notepad++; no Linux, o Nano Editor.

# OBJETIVOS

## MÓDULO 1

Identificar os fundamentos da CSS

## MÓDULO 2

Reconhecer os recursos de cores, texto, fontes e webfontes da CSS3

## MÓDULO 3

Identificar os conceitos de Box Model, pseudoclasses, pseudo-elementos e posicionamento

## MÓDULO 4

Reconhecer Frameworks CSS

# INTRODUÇÃO

A CSS, ou Folhas de Estilo em Cascata (Cascading Style Sheets), é uma linguagem de estilo que fornece total controle sobre a apresentação de um documento escrito em HTML.

No ambiente Web, a HTML é a linguagem responsável pela estrutura do conteúdo de uma página. Embora seja capaz também de organizar o conteúdo visualmente, é função da CSS cuidar desse aspecto e de tudo relacionado ao estilo e layout da página.

Com a CSS, é possível, por exemplo, alterar a forma e posicionamento dos elementos, as cores, tipos e tamanhos de fontes, e muito mais.

# MÓDULO 1

---

🕒 Identificar os fundamentos da CSS

## COMO A CSS FUNCIONA?

A CSS, ou Folhas de Estilo em Cascata (*Cascading Style Sheets*), é uma linguagem de estilo que fornece total controle sobre a apresentação de um documento escrito em HTML. Com ela, é possível, por exemplo, alterar a forma e posicionamento dos elementos, as cores, tipos e tamanhos de fontes e muito mais.

A CSS permite a aplicação seletiva de estilos a elementos em uma página HTML. Isso significa dizer que um ou mais estilos podem ser aplicados em um documento inteiro ou mesmo em apenas parte dele. Além disso, um mesmo tipo de elemento pode ter, ao longo do documento, diferentes estilos.



# SINTAXE DA CSS

Para aplicar um estilo CSS a um elemento específico, é necessário identificá-lo e apontar qual de suas propriedades queremos alterar e qual valor queremos atribuir-lhe. Essas três informações definem a sintaxe da CSS, conforme pode ser visto no exemplo a seguir.



Fonte: Elaborado pelo autor.

📷 Figura 1: Sintaxe da CSS.

Tendo o exemplo anterior como base, podemos perceber que, para aplicarmos um estilo utilizando CSS, são necessários:

O **seletor**: nesse caso, a tag HTML `<p>`;

Ao menos uma **propriedade**: a cor de fundo (`background-color`);

Ao menos um **valor** para a propriedade: `blue`.

Essa declaração de estilo faria com que todas as tags `<p>` do documento apresentassem a cor azul ao fundo. O exemplo utilizou apenas uma declaração (propriedade + valor), mas é possível inserir em conjunto várias outras.

Além dos aspectos mencionados acima, há outros importantes quanto à sintaxe:

A propriedade e seu valor devem ser separados por dois pontos `“:”`;



Uma declaração deve ser separada da declaração subsequente com a utilização do ponto-e-vírgula `“;”`;



O conjunto de estilos aplicados a um seletor é envolvido por chaves “{” e “}”.

Vamos ver a seguir um exemplo de duas propriedades dadas à tag <p> e o resultado dessas declarações no navegador.

```
p {  
    background-color: blue;  
    color: white;  
}
```

Texto do parágrafo estilizado com CSS

Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 2: Demonstração da aplicação de estilos.

## SELETORES

Vimos nos exemplos anteriores um estilo sendo declarado em uma tag HTML <p>. Nós nos referimos a essa tag como sendo o seletor ao qual o estilo foi aplicado. Há muitos outros seletores disponíveis além daqueles correspondentes às tags HTML, conforme veremos a seguir.

## SELETORES CLASS E ID

O seletor de classe é definido a partir da declaração do atributo “class” em um elemento HTML. Já o de identificação é definido com o atributo “id”.

Importante frisar que, embora um elemento possa ter mais de uma classe, terá somente um identificador.

Em termos de nomenclatura para a definição do nome da classe ou do identificador, não existe uma regra a ser seguida. Procure utilizar nomes que façam sentido, que tenham relação com a função do elemento na página e que, de fato, ajude a identificá-lo ou classificá-lo.

Logo abaixo, podemos ver um exemplo de sintaxe correspondente à declaração desses dois atributos na HTML e na CSS.

Na primeira parte da imagem, é apresentado um fragmento de código HTML. Repare que uma mesma classe, a “texto vermelho”, foi atribuída à tag <h1> e a uma das tags <p>. Com isso, vemos que uma classe pode ser atribuída a mais de um elemento. Agora note a sintaxe para atribuição de múltiplas classes a um elemento na segunda tag <p>, à qual foram atribuídas as classes “texto\_descricao” e “texto\_vermelho”.

Em relação ao código CSS, veja que o seletor de ID é representado por uma “#” e o de class por um “.”, seguidos de seus nomes. Além disso, foram apresentadas duas formas de sintaxe que produzirão o mesmo efeito. A diferença entre elas é que, na segunda, o nome da tag à qual a identificação ou classe foi atribuída precede o respectivo sinal.

#### Fragmento HTML

```
...
<h1 class="texto_vermelho">Titulo Principal</h1>
<p id="texto_apresentacao">
  Texto do parágrafo com atributo de identificação.
</p>

<h2>Primeiro Subtitulo</h2>
<p class="texto_descricao texto_vermelho">
  Texto do parágrafo com atributo de classe.
</p>

<h2>Segundo Subtitulo</h2>
<p class="texto_descricao">
  Texto do parágrafo com atributo de classe.
</p>
...
```

#### Sintaxe 1

```
<style type="text/css">
  #texto_apresentacao{
    font-size:16px;
  }

  .texto_descricao{
    font-size:12px;
  }

  .texto_vermelho{
    color:red;
  }
</style>
```

#### Sintaxe 2

```
<style type="text/css">
  p#texto_apresentacao{
    font-size:16px;
  }

  p.texto_descricao{
    font-size:12px;
  }

  .texto_vermelho{
    color:red;
  }
</style>
```

Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 3: Sintaxe de declaração de seletores de classe e identificação.

## RESTRIÇÕES E BOAS PRÁTICAS NA UTILIZAÇÃO DO IDENTIFICADOR

Embora não exista um padrão ou preferência quanto a utilizar o seletor “id” ou “class”, é importante frisar novamente que um “id” deve ser aplicado a apenas um elemento, enquanto a

“class” pode ser aplicada a um ou vários elementos.

Embora o navegador não faça uma verificação se um mesmo “id” foi utilizado em diferentes elementos, tal método pode trazer alguns problemas de estilização e comportamento, uma vez que esse seletor também é bastante usado pelo Javascript. Frente a isso, adote a boa prática de definir identificadores únicos.

## SELETORES DE ATRIBUTO

Esses seletores utilizam nomes de atributos dentro de colchetes, sendo possível combiná-los com valores. Abaixo são mostrados alguns dos seletores de atributo disponíveis:

**[checked]** - Seleciona todos os elementos que possuem o atributo “checked”;

**[type='text']** - Seleciona todos os elementos do tipo “text”.

Os seletores de atributo são bastante flexíveis, permitindo inúmeras combinações. Por exemplo, é possível usá-los para selecionar todas as imagens com uma determinada extensão, selecionar todos os elementos com o atributo title contendo determinado valor etc.

## SELETORES BASEADOS EM RELACIONAMENTO

É possível declarar estilos utilizando a relação entre os elementos. A tabela a seguir mostra os principais seletores baseados em relacionamento.

Seletor	Seleção
H1 P	Qualquer elemento P que seja descendente (filho, neto etc.) de um elemento H1.
H1 > P	Qualquer elemento P que seja filho de um elemento H1.

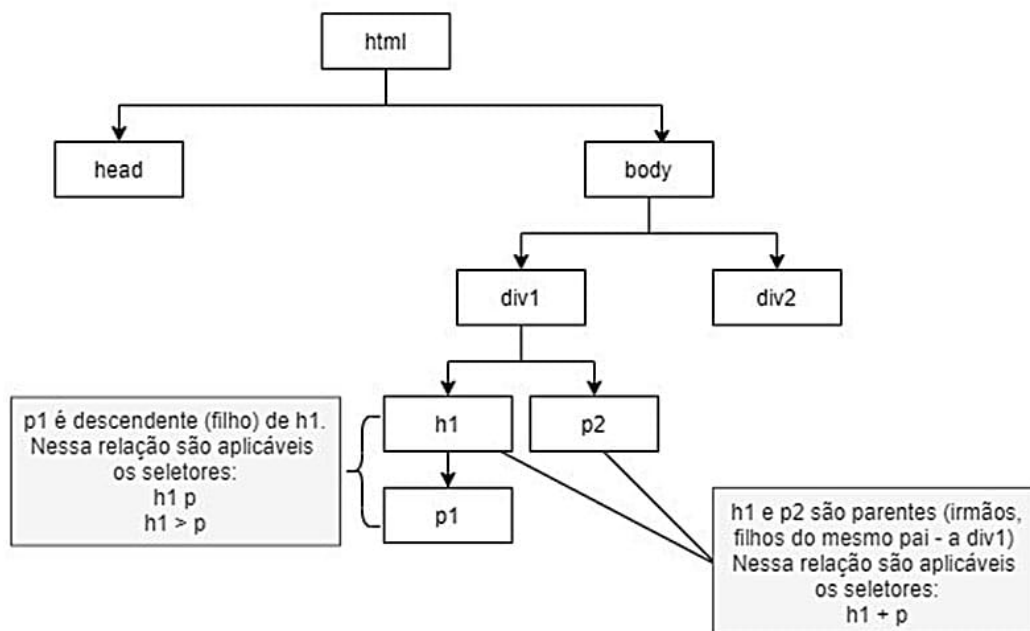
H1 + P

Qualquer elemento P que seja o próximo irmão de um elemento H1 (ou seja, o próximo filho de um mesmo pai).

**Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Tabela 1: Principais seletores baseados em relacionamento.

Para uma melhor compreensão quanto à descendência e parentesco mencionados acima, veja a figura abaixo. Essa é uma representação simbólica da Árvore DOM (representação estruturada do documento HTML em formato de árvore) contendo elementos representando tags HTML. Veja ainda a explicação de cada relação e aplicação dos seletores baseados em relacionamento.



Fonte: Elaborado pelo autor.

📷 Figura 4: Relação entre elementos em uma página HTML.

## PROPRIEDADES

Existem inúmeras propriedades CSS, desde as definidas pela sua especificação, ditas **propriedades padrão**, até as **proprietárias**, que funcionam apenas em alguns navegadores. A fim de garantir uma maior compatibilidade, assim como otimizar o desenvolvimento, deve-se



sempre dar preferência às primeiras. A seguir são apresentadas algumas das propriedades mais comuns da CSS.

Propriedade	Função
Background	Estilizar o fundo de elementos. Para tal há uma série de propriedades, além do atalho “Background”, como “background-color”, “background-image” etc.
Border	Controla as bordas de um elemento, sendo possível definir suas cores, espessuras, entre outras propriedades.
Top, Bottom, Right e Left	Controlam o posicionamento, relativo ou absoluto, dos elementos em relação a outros elementos.
Color	Estila a cor do conteúdo textual de um elemento.
Font-family, Font-size, Font-weight, etc.	Há uma série de propriedades para estilizar o conteúdo textual de um elemento no que diz respeito à fonte, como a família de fontes, seu tamanho, peso (mais clara ou mais escura - negrito) etc.
Height	Definir a altura de um elemento.
List-style, List-style-image, etc.	Há algumas propriedades para estilizar as listas HTML.

Margin	Controla a distância em função da margem de um elemento para outro.
Padding	Controla a distância entre as bordas e o conteúdo de um elemento.
Position	Define como um elemento deve ser posicionado na página.
Text-...	Muitas propriedades controlam o comportamento do conteúdo textual de um elemento, como alinhamento (justificado, centralizado etc.), aparência (sublinhado etc.) etc.
Width	Definir a largura de um elemento.
Z-index	Definir a profundidade de um elemento – usado, por exemplo, para sobreposição de elementos.

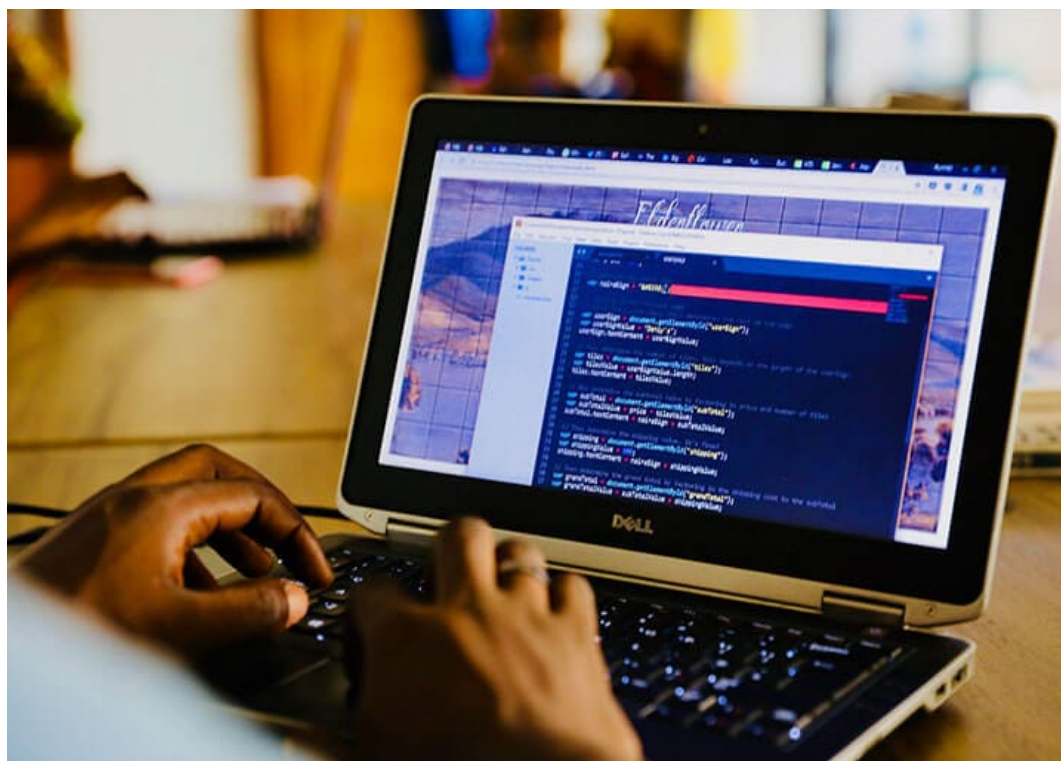
**Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Tabela 2: Lista de propriedades CSS mais comuns.



# INTEGRANDO A CSS À HTML

Há três formas usuais de aplicar estilos em um documento HTML fazendo uso de CSS: Inline, Interna e Externa. Além dessas, a HTML5 permite ainda a aplicação em Escopo. Vamos conhecer um pouco mais de cada uma delas?



Fonte: Unsplash

## CSS INLINE

Essa forma implica em declarar o estilo CSS diretamente na tag, no código HTML. Veja a seguir um exemplo de um estilo apresentado anteriormente sendo aplicado de forma Inline.



Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 5: Aplicação de estilo Inline.

A declaração Inline faz uso do atributo “style” precedido por declarações separadas por ponto-e-vírgula “;”. Esse atributo pode ser usado em qualquer tag HTML.

## CSS INTERNA

Também chamada de CSS Incorporada, é declarada na seção <head> do documento HTML.



Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 6: Aplicação de estilo CSS interna.

A declaração Inline faz uso do atributo “style” precedido por declarações separadas por ponto-e-vírgula “;”. Esse atributo pode ser usado em qualquer tag HTML.

## CSS EXTERNA

Nesse caso, os estilos são declarados em um arquivo externo, com extensão “.css” e vinculados ao documento HTML por meio da tag <link> ou da diretiva @import dentro da tag <head>. Ambos os exemplos podem ser vistos logo a seguir:



Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 7: Aplicação de Estilo Externa.

## CSS EM ESCOPO

Essa forma de aplicação de estilo foi criada a partir da HTML5. Com ela, é possível aplicar estilos no âmbito de escopo, ou seja, específicos para as seções da página onde foram declarados, incluindo os seus elementos filhos. No código abaixo, a tag <p> receberá os estilos definidos, sendo a mesma regra válida para outros estilos e elementos que, porventura, venham a fazer parte da <div>.



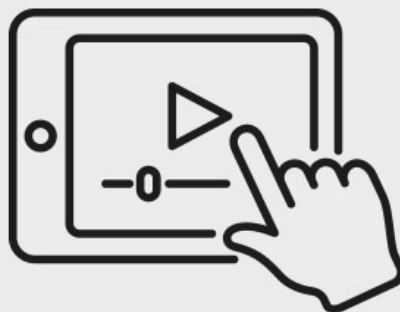
Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 8: Aplicação de estilo a nível de Escopo.



## INTEGRANDO A CSS À HTML

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



## EFEITO CASCATA

Quando trabalhamos com CSS, é comum nos depararmos com a declaração conflitante de estilos, ou seja, diferentes definições de estilo para um mesmo elemento. Nessas situações, entra em ação o Efeito Cascata. Para entendermos a definição desse efeito, é preciso abordarmos outros dois conceitos: **herança** e **especificidade**.


# HERANÇA

O CSS permite que a aplicação de propriedades a elementos pais seja herdada pelos seus elementos filhos. Peguemos como exemplo o código abaixo.

```
...
div{
  color: blue;
}
...
<div>
  Texto solto na DIV.
  <p>Texto do parágrafo que é "filho" da DIV</p>
</div>
...
```



Fonte: Elaborado pelo autor no Notepad++.

 Figura 9: Exemplo de herança em CSS.

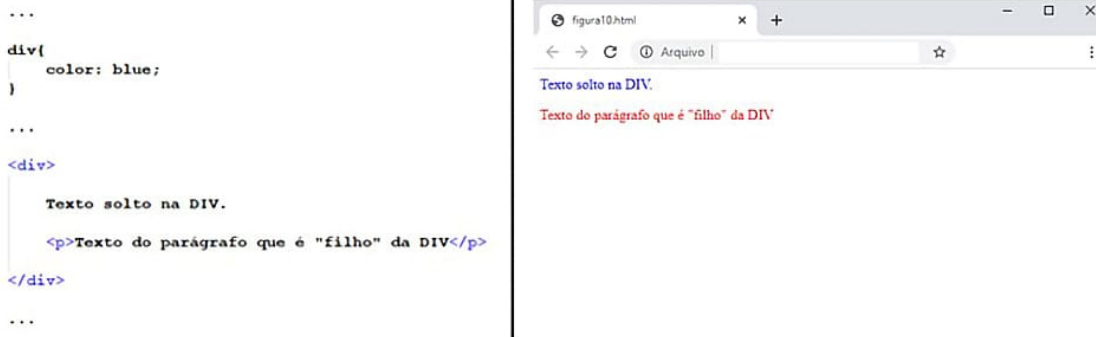
O resultado do fragmento de código mostrado anteriormente mostrará tanto o texto solto quanto o texto dentro da tag <p> com a cor azul. Isso significa que a tag <p> herdou o estilo definido para o seu pai, a tag <div>.

Essa capacidade de herança de estilos caracteriza o que chamamos de Efeito Cascata.

Cabe destacar que nem todas as propriedades CSS podem ser herdadas pelos filhos. Um exemplo são as propriedades relacionadas à formatação das boxes (que veremos mais adiante), como largura, altura, entre outras.

## ESPECIFICIDADE

Para entender o que é a especificidade no âmbito das folhas de estilo, vamos recorrer a mais um exemplo.



Fonte: Elaborado pelo autor no Notepad++.

 Figura 10: Exemplo de especificidade em CSS.

Perceba que o fragmento da Figura 9 foi adaptado. Antes, era aplicado o conceito de herança, assim o texto dentro da tag filha assumia o estilo definido para o seu pai. Agora, há um estilo específico definido para todas as tags `<p>` que sejam filhas de tags `<div>`. Com isso, ao visualizarmos o resultado no navegador, teremos o texto solto na cor azul e o texto dentro da tag `<p>` na cor vermelha.

Esse foi um exemplo simples. A CSS é bastante flexível e nos permite definir diferentes níveis de especificidade. Entretanto, é importante termos cuidado com a sobreposição de estilos – diferentes estilos definidos para um mesmo elemento em diferentes partes de nosso código CSS. A regra, nesse caso, é: prevalecerá o estilo mais específico. No exemplo acima, a primeira declaração (para a tag `div`) é generalizada; a segunda (`div p`), específica.

## DICAS SOBRE AS REGRAS DE PRECEDÊNCIA

A regra de precedência em relação às formas de inclusão da CSS segue a seguinte ordem:

Os estilos Internos e de escopo têm precedência sobre estilos em arquivos externos;

Os estilos Inline têm precedência sobre estilos internos, de escopo e externos.

Quanto aos seletores, a regra de precedência segue a seguinte ordem:

Seletores de elemento (utilização apenas do nome da tag) são os de menor precedência, por serem muito genéricos;



Seletores de classe têm mais precedência que os de elemento;



Seletores de identificação têm mais precedência que os de classe.

## !IMPORTANT

Veja este novo exemplo:

```
...

p{
    color: blue!important;
}

div p{
    color:red;
}

...

<div>
    Texto solto na DIV.
    <p>Texto do parágrafo que é "filho" da DIV</p>
</div>

...
```

Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 11: Exemplo de aplicação do valor !important.

Seguindo as regras de Especificidade, sua aplicação resultaria na apresentação do texto dentro da tag <p> na cor vermelha, pois sua declaração de estilo é mais específica que a utilizada para a tag <p> (texto azul), que é generalizada. Entretanto, a utilização do valor !important, que se enquadra no que chamados de **css hack**, na declaração mais generalizada, faz com que esse estilo se sobreponha ao específico. Logo, o código acima resulta na apresentação do texto dentro da tag <p> na cor azul.



# CSS HACK

Técnica de codificação usada para alterar o comportamento natural da CSS e para ocultar ou mostrar uma determinada declaração de acordo com o navegador, sua versão ou recursos disponíveis.

## CSS3

A CSS, atualmente, está em sua terceira versão, a CSS3. Dentre as diversas novidades dessa versão, destacam-se:

Melhorias nos seletores, com novas possibilidades de seleção: primeiro e/ou último elemento, elementos pares ou ímpares etc.;

Efeito gradiente e de sombra em textos e elementos;

Bordas arredondadas;

Manipulação de opacidade;

Controle de rotação e perspectiva;

Animações.



Fonte: Pixabay

## VERIFICANDO O APRENDIZADO

### 1. A RESPEITO DA INTEGRAÇÃO HTML E CSS, ASSINALE A AFIRMATIVA CORRETA:

- A)** Tanto a HTML quanto a CSS são renderizadas pelo navegador que, interpretando as tags de marcação e os estilos que lhes são aplicados, as exibe em tempo de execução/requisição pelo usuário.
- B)** Todo o código CSS é compilado pelo servidor web que o transforme em código HTML nativo a fim de que possa ser exibido no navegador.
- C)** A CSS inline, incorporada e de escopo são renderizadas diretamente pelo navegador, juntamente com a HTML. Já a CSS externa, por não estar dentro do arquivo HTML, precisa ser compilada pelo servidor web antes de ser renderizada.
- D)** Apenas a partir da HTML5, com a possibilidade de declaração de estilos em escopo, os navegadores passaram a dar suporte à renderização da CSS e do HTML sem necessidade de

compilação.

**2. SOBRE A ESPECIFICIDADE, ASSINALE A OPÇÃO QUE CORRESPONDE AO ESTILO MAIS ESPECÍFICO E QUE, CONSEQUENTEMENTE, SERÁ APLICADO AO ELEMENTO < P > ABAIXO:**

**< DIV >**

**< P ID = "IDENTIFICADOR" CLASS = "CLASSE" >  
TEXTO DO PARÁGRAFO.**

**< /P >**

**< /DIV >**

**A) div > p { background-color: blue; }**

**B) #identificador{ background-color: black; }**

**C) p#identificador{ background-color: red; }**

**D) p.classe{ background-color:pink; }**

---

## **GABARITO**

**1. A respeito da integração HTML e CSS, assinale a afirmativa correta:**

A alternativa **"A "** está correta.

Tanto a HTML quanto a CSS são linguagens interpretadas diretamente pelo browser e que não precisam ser compiladas – exceto a CSS quando se utiliza pré-processadores.

**2. Sobre a especificidade, assinale a opção que corresponde ao estilo mais específico e que, consequentemente, será aplicado ao elemento < p > abaixo:**

**< div >**

**< p id = "identificador" class = "classe" >  
Texto do parágrafo.**

**< /p >**

**< /div >**

A alternativa **"C "** está correta.

As regras que utilizam seletores têm maior precedência. Entretanto, quanto mais específico, maior a precedência. Logo, a opção c é mais específica que a opção b).

## MÓDULO 2

---

### ⦿ Reconhecer os recursos de cores, texto, fontes e web fontes da CSS3

Até aqui, vimos o conceito de CSS, sua sintaxe e a forma de integração em páginas HTML. Além disso, vimos algumas propriedades mais comuns foram apresentadas. Neste módulo, aprofundaremos o estudo das propriedades relacionadas a cores, texto, fontes e web fontes, tomando como base recente especificação CSS, a CSS3.

## CORES

Com a utilização de CSS, podemos manipular as cores de elementos HTML, seja na aparência das caixas, seja na cor de texto. Para isso, há uma série de propriedades CSS disponíveis para diversos elementos, mas antes vamos abordar as formas de definição de cores.

## FORMAS DE ESCRITA DE CORES

As cores em CSS podem ser escritas de três modos:

Com palavras-chave, onde podem ser usados os nomes das cores (seguindo as definidas pela especificação CSS) ou a notação hexadecimal. Ex.: blue, red, #FFFFFF etc;

Com um sistema de coordenada cúbica RGB, com as notações rgb() e rgba();

Com um sistema de coordena cilíndrica HSL, com as notações hsl() e hsla().

## PROPRIEDADES DE COR

Em quais elementos podemos definir cores.

Veja na tabela a seguir as principais, mas não as únicas, propriedades relacionadas à cor, bem como os elementos aos quais podem ser aplicadas.

Propriedade	Serve para definir	Onde pode ser utilizada
color	cor de textos	Elementos que contenham texto, como <h1> ... <h6>, <p>, <header>, <section>, etc.
background-color	cor de fundo de elementos	Aplica-se a qualquer elemento HTML.
border-color	cor da borda	Aplica-se a qualquer elemento HTML.
outline-color	cor da borda externa	Aplica-se a qualquer elemento HTML.

**Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Tabela 3: Lista das propriedades de cor e elementos onde podem ser aplicadas.

# TEXTO

A estilização de textos com o uso de CSS é dividida em duas partes:

Em linhas gerais, os navegadores aplicam estilos padrões quando renderizam conteúdos textuais. A seguir, serão vistas algumas propriedades CSS que alteram esse comportamento padrão.

O layout do texto – ou seja, espaçamento entre os caracteres e linhas; alinhamento em relação ao *container*.



Estilos das fontes – família, tamanho, efeitos como negrito etc.

Em linhas gerais, os navegadores aplicam estilos padrões quando renderizam conteúdos textuais. A seguir, serão vistas algumas propriedades CSS que alteram esse comportamento padrão.

## ALINHAMENTO DE TEXTO

A propriedade `text-align` é usada para controlar o alinhamento do texto em razão do contêiner no qual está inserido.

Tal propriedade pode assumir 4 valores: `left`, `right`, `center` e `justify`. Como os nomes indicam, essas propriedades alinham o texto à esquerda, direita, ao centro ou de forma justificada.

## ESPAÇAMENTO ENTRE LINHAS

A propriedade `line-height` permite alterar o espaçamento vertical entre as linhas de texto. Seus valores possíveis são:

**Normal** – valor padrão do navegador (entre 1 e 1.2 em relação ao `font-size`, dependendo do navegador);

**Número** – valor inteiro ou decimal que será multiplicado ao tamanho da fonte;

**Comprimento** – valor unidades como pixels, pontos, “em” etc.

A maneira mais recomendada de declarar o espaçamento entre linhas é utilizando o valor em número. Nele, o espaçamento será o resultado da multiplicação do valor definido pelo tamanho da fonte. Por exemplo: `line-height: 1.5`; `font-size: 12px`; onde valor 1.5 será multiplicado pelo valor da propriedade `font-size`, resultando no valor de “18 px” de espaçamento.

## ESPAÇAMENTO ENTRE LETRAS E PALAVRAS

As propriedades `letter-spacing` e `word-spacing` permitem alterar o espaçamento entre letras e/ou palavras. Elas podem assumir valores de comprimento – “px”, “pt” etc.

## FONTES

Em relação às fontes, há propriedades CSS para definir família, tamanho, estilo, entre outras possibilidades. Vamos conhecer as propriedades mais usadas?

### FONT-FAMILY

Essa propriedade é utilizada para definir a família da fonte utilizada em página web ou em partes de seu conteúdo. Utilizando-a, é possível definir, por exemplo, desde uma única fonte a uma lista de fontes, onde os seus valores são declarados em ordem de importância, da esquerda para direita. Desse modo, caso uma determinada fonte não esteja disponível no dispositivo cliente, a próxima definida será usada, e assim sucessivamente. Caso nenhuma das fontes definidas estejam disponíveis no cliente, o navegador fará uso de uma fonte padrão.

Estes são exemplos de famílias de fonte: Arial, Verdana, “Times New Roman” (fontes com nomes compostos devem ser declaradas utilizando-se aspas), entre outras. As fontes citadas anteriormente e algumas outras formam o conjunto de fontes chamadas de Fontes Seguras para Web (*Web Safe Fonts*). Sempre que possível, deve-se dar preferência à utilização dessas fontes “seguras”, pois possuem suporte pela maioria dos sistemas operacionais.

### FONT-SIZE

A propriedade `font-size` é responsável por definir o tamanho do texto. Seus valores podem ser definidos com a utilização de diferentes unidades de medida, como pixels, porcentagem, “em” etc.

## FONT-STYLE

Propriedade usada na estilização de textos aplicando o efeito de itálico. Seus valores possíveis são: normal (ou seja, tira o efeito do texto, sendo o estilo padrão de todo elemento), italic e oblique (uma versão mais inclinada em relação ao itálico).

## FONT-WEIGHT

O peso de uma fonte é definido com a utilização dessa propriedade. Com ela, é possível aplicar o efeito de negrito em uma escala. Seus valores possíveis são: normal, bold, lighter e bolder (aumentam ou diminuem o peso da fonte em relação ao peso da fonte de seu elemento pai); e uma escala numérica que vai de 100 a 900.

Há algumas boas práticas e cuidados a serem levados em consideração quando se trabalha com estilização de fontes usando CSS. Uma delas diz respeito ao controle sobre a possível degradação que pode ocorrer na página. Portanto, deve-se tomar os devidos cuidados optando pela utilização de uma lista de fontes e mantendo por último as fontes “genéricas”, como Serif, “Sans-Serif” e Monospace. Desse modo, haverá maior garantia e controle sobre o que o usuário verá como resultado.

## WEB FONTES

As Web Fontes são um importante recurso em termos de tipografia. Se antes a sua estilização ficava restrita àquelas disponíveis nos sistemas operacionais dos usuários, a partir da implementação da regra `@font-face` tornou-se possível a utilização de Web Fontes. Essa nova propriedade permite a utilização de fontes que, ao serem definidas, são baixadas pelo navegador no momento de carregamento da página. Logo, sua utilização permite um controle maior do layout de uma página no que diz respeito às fontes, além da possibilidade de serem usadas fontes com maior apelo visual.

## COMO UTILIZAR A REGRA @FONT-FACE



A declaração da regra `@font-face` é feita pela definição de duas principais propriedades: **font-family** e **src**. Na primeira, definimos um nome para a família da fonte que estamos importando, usando-o ao longo do arquivo CSS. A segunda aponta para a url onde o arquivo da fonte se encontra. Vamos ver a seguir a fonte Awesome sendo declarada.

```
@font-face {
  font-family: 'Awesome';
  font-style: normal;
  font-weight: 400;
  src: local('Awesome Font'),
       url('/assets/fontes/awesome.woff2') format('woff2'),
       url('//siteondeafonteestadisponivel/fonts/awesome.woff') format('woff'),
       url('/assets/fontes/awesome.ttf') format('truetype'),
       url('//outrositeondeafonteestadisponivel/fonts/awesome.eot') format('embedded-opentype');
}
```

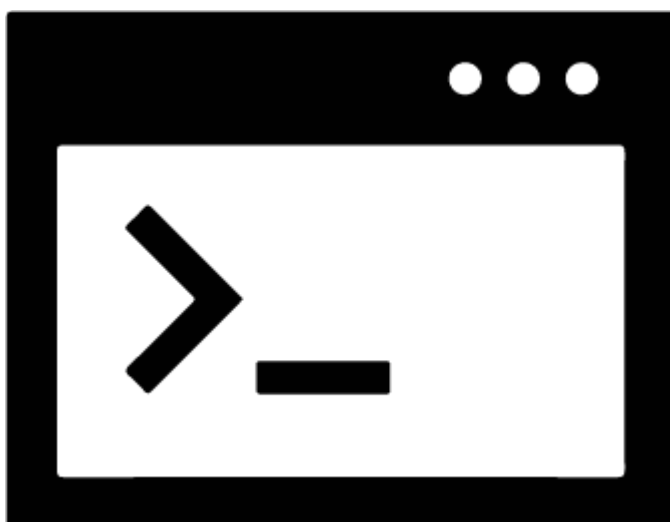
Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 12: Declaração de Web Fonte.

Como podemos observar, além das propriedades `font-family` e `src`, há outras que podem ser aplicadas às Web Fontes. Em relação à `font-family`, a partir do momento da sua declaração, o nome definido poderá ser utilizado para estilizar qualquer outro elemento ao longo do CSS – considere que podemos tanto utilizar uma única família de fontes para o documento HTML inteiro como a combinação de diferentes famílias.

O código também mostra que as fontes incorporadas podem tanto estar hospedadas localmente quanto na internet. Além disso, há alguns outros elementos na declaração dos quais ainda não falamos. São eles: as funções **local** e **format**.

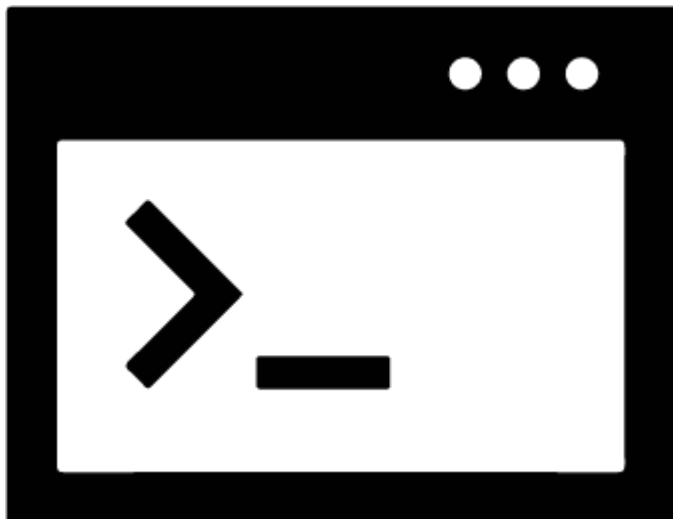
Cabe destacar também os diferentes tipos existentes de fontes web. Aprofundaremos esses elementos extra a seguir.



Ícone do word

# A FUNÇÃO LOCAL

Essa função indica ao navegador que, antes de fazer o download da fonte definida, deverá verificar se ela já está disponível na máquina do usuário.



Ícone do word

# A FUNÇÃO FORMAT

Também chamada de dica, essa função opcional é utilizada quando se deseja declarar vários formatos de fontes, indicando justamente o formato de cada uma. No exemplo acima, temos os formatos “woff”, “woff2”, “ttf” e “eot”.

# TIPOS DE WEB FONTES

Estão disponíveis atualmente alguns tipos de Web Fontes, cuja escolha deve considerar fatores como a compatibilidade com a maioria dos navegadores e o tamanho dos arquivos. Veja a seguir os tipos mais comuns de web fontes.

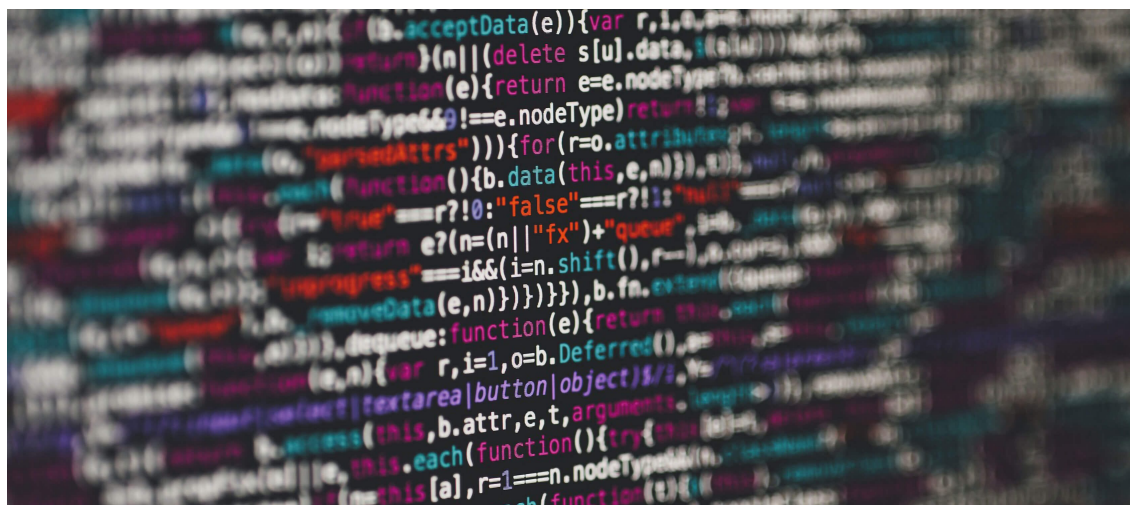
Tipo	Navegadores x Versões (iniciais com suporte)				

	Google Chrome	Firefox	Internet Explorer / Edge	Safari	Opera
TTF/OTF	4.0	3.5	9.0	3.1	10.0
WOFF	5.0	3.6	9.0	5.1	11.1
WOFF2	36.0	35.0	--	--	26.0
SVG	4.0	--	--	3.2	9.0
EOT	--	--	6.0	--	--

**Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Tabela 4: Tipos mais comuns de Web Fontes.

Como visto na tabela, alguns tipos oferecem melhor suporte em relação aos navegadores mais atuais. Entretanto, não dão suporte às versões antigas. Isso reforça a recomendação anterior: considere sempre utilizar diferentes fontes para oferecer uma melhor experiência aos usuários.



# ABREVIATURAS OU ATALHOS EM CSS

As Folhas de Estilo permitem a aplicação de algumas propriedades utilizando abreviaturas ou atalhos. O exemplo a seguir mostra as duas formas de realizar uma mesma declaração:

<pre>{ 1 p{ 2   margin-top: 10px; 3   margin-bottom: 8px; 4   margin-right: 6px; 5   margin-left: 4px; } 6 }</pre>	<pre>{ 1 p{ 2   margin: 10px 6px 8px 4px; } 3 }</pre>
--	---

Fonte: Elaborado pelo autor no Notepad++.

---

<pre>{ 1 p{ 2   padding-top: 10px; 3   padding-bottom: 8px; 4   padding-right: 6px; 5   padding-left: 4px; } 6 }</pre>	<pre>{ 1 p{ 2   padding: 10px 6px 8px 4px; } 3 }</pre>
--	--

Fonte: Elaborado pelo autor no Notepad++.

---

<pre>{ 1 p{ 2   border-width: 2px; 3   border-style: solid; 4   border-color: #cccccc; } 5 }</pre>	<pre>{ 1 p{ 2   border: 2px solid #cccccc; } 3 }</pre>
--	--

Fonte: Elaborado pelo autor no Notepad++.

---

<pre>{ 1 p{ 2   background-color: #000000; 3   background-image: url(imagem.jpg); 4   background-repeat: no-repeat; 5   background-position: top left; } 6 }</pre>	<pre>{ 1 p{ 2   background: #000000 url(imagem.jpg) no-repeat top left; } 3 }</pre>
--	---

Fonte: Elaborado pelo autor no Notepad++.

---

<pre>{ 1 p{ 2   font-size: 1em; 3   line-height: 1.5em; 4   font-weight: bold; 5   font-style: italic; 6   font-family: verdana; } 7 }</pre>	<pre>{ 1 p{ 2   font: 1em/1.5em bold italic verdana; } 3 }</pre>
--	--

Fonte: Elaborado pelo autor no Notepad++.

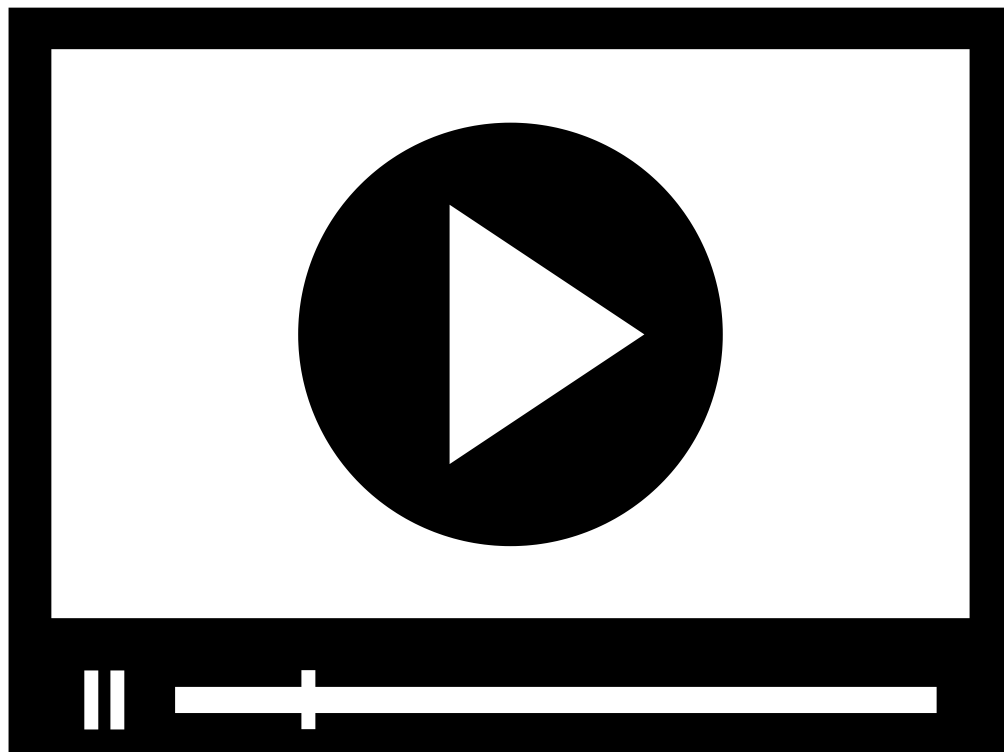
---

```

{ 1 ul{
  2   list-style: #000000;
  3   list-style-type: disc;
  4   list-style-position: outside;
  5   list-style-image: url(imagem.jpg);
  6 }
}

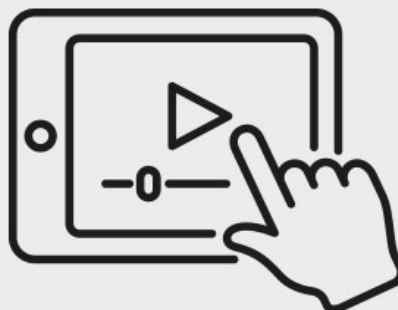
```

Fonte: Elaborado pelo autor no Notepad++.



## FONTES

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



## VERIFICANDO O APRENDIZADO

## **1. SOBRE A ESTILIZAÇÃO DE TEXTOS E FONTES, OS NAVEGADORES POSSUEM ESTILOS PADRÕES PARA ESSES TIPOS DE ELEMENTO. LOGO, É CORRETO DIZER QUE:**

- A)** Os estilos aplicados por padrão pelos navegadores existem para permitir que o controle do layout do conteúdo da página fique nas mãos do usuário, e não do desenvolvedor.
- B)** Os navegadores padronizam os estilos dos elementos de texto e fonte para garantirem a usabilidade e acessibilidade das páginas.
- C)** A CSS permite total controle sobre os elementos de texto e fonte. Com isso, todo o controle fica nas mãos do desenvolvedor, que poderá alterar qualquer aspecto desses elementos, tornando assim a página uniforme, uma vez que não dependerá dos estilos padrão dos navegadores, que são diferentes entre si.
- D)** Embora a CSS permita a estilização de textos e fontes, os navegadores sempre terão controle sobre o layout da página, podendo, inclusive, redefinir os estilos CSS que não estejam de acordo com os padrões de acessibilidade.

## **2. ASSINALE A AFIRMATIVA CORRETA QUANTO À UTILIZAÇÃO DE WEB FONTES EM RELAÇÃO ÀS FONTES CSS PADRÕES:**

- A)** Por serem mais leves, uma vez que são nativas, as fontes definidas através de CSS sempre serão renderizadas, sem qualquer tipo de restrição, em qualquer sistema operacional.
- B)** As web fontes devem ser usadas, em detrimento das fontes padrão, por terem maior apelo visual.
- C)** A melhor escolha em relação aos estilos de fontes é não usar nem fontes padrão e nem webfontes, ou seja, é deixar que fique a cargo do navegador escolher a fonte padrão de acordo com as disponíveis no sistema operacional do usuário.
- D)** Além de fornecerem mais opções, em termos visuais, as web fontes, quando usadas adequadamente, garantem uma menor degradação das páginas, uma vez que não haverá dependência do ambiente do usuário, quanto a esse possuir ou não a fonte definida.

---

## GABARITO

**1. Sobre a estilização de textos e fontes, os navegadores possuem estilos padrões para esses tipos de elemento. Logo, é correto dizer que:**

A alternativa **"C "** está correta.

A CSS permite total controle sobre qualquer elemento em uma página. Deve-se ter em mente, ao utilizá-la, não só as preocupações com estética, mas também com usabilidade e acessibilidade, garantindo assim a melhor experiência possível aos usuários.

**2. Assinale a afirmativa CORRETA quanto à utilização de web fontes em relação às fontes CSS padrões:**

A alternativa **"D "** está correta.

As web fontes permitem um maior controle visual sobre como cada usuário verá o site, diminuindo assim a dependência de fatores externos, como a disponibilidade de fontes no computador do visitante.

## MÓDULO 3

---

🕒 Identificar os conceitos de Box Model, pseudoclasses, pseudo-elementos e posicionamento

## BOX MODEL

Nos módulos anteriores, vimos os conceitos básicos de CSS, sua sintaxe, seus elementos e suas formas de integração com HTML. Abordamos também como aprimorar o design de páginas com os estilos de cores, texto e fontes.

Neste módulo, avançaremos um pouco mais e percorreremos os conceitos de Box Model (modelo de caixas ou retângulos), Pseudoclasses e Pseudo-elementos e Posicionamento.



Os elementos de uma página web são representados por uma caixa que possui o formato de um retângulo. Uma boa analogia aqui seria compararmos a marcação HTML a brinquedos de bloco.



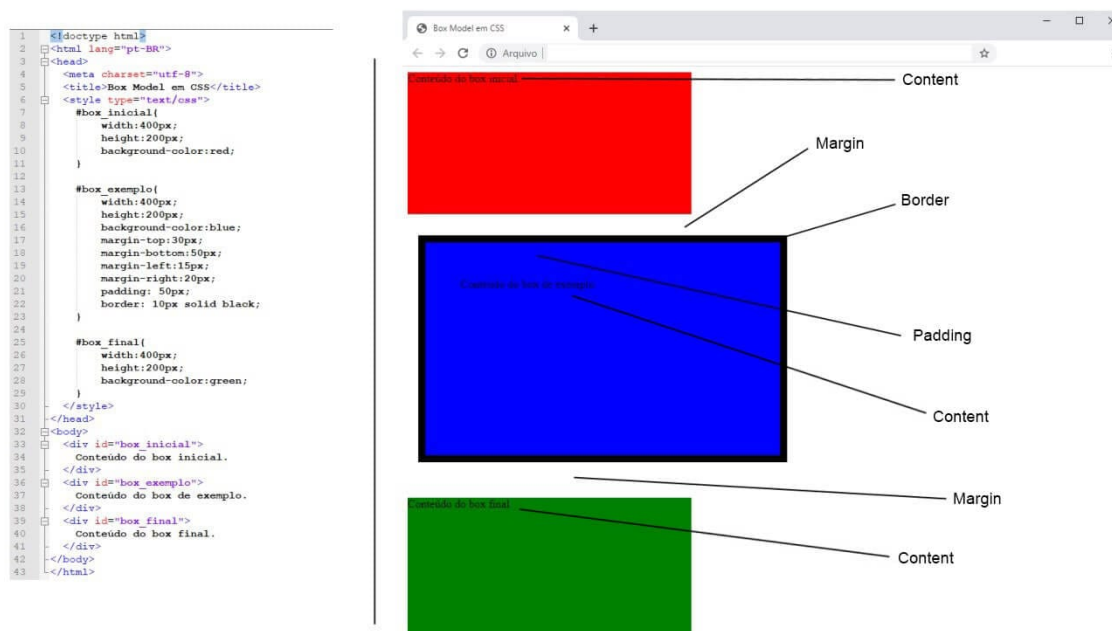
No Lego, por exemplo, há peças de diferentes tamanhos, larguras, alturas, cores e formatos. Tendo em mãos tais elementos, é nosso papel montá-los, encaixá-los, de forma harmoniosa para, ao final, obtermos o resultado esperado.





Para chegar a esse resultado, é importante entendermos o comportamento, a composição e as características de cada bloco, assim como os estilos que podem receber.

Nesse sentido, dentro do conceito de Box Model – que, em CSS, está relacionado a design e layout – nossos boxes possuem quatro componentes principais: margem (margin), borda (border), preenchimento (padding) e conteúdo (content).



Fonte: Elaborado pelo autor no Notepad++.

📷 Figura 13: Componentes do Box Model.

A imagem anterior nos ajuda a compreender a composição do Box Model de maneira prática. Repare que foram definidas 3 caixas com o elemento `<div>` e, para cada uma delas, foram definidos diferentes estilos, como largura, altura e cor de fundo. Para a `<div>` com identificador “box\_exemplo” foram declarados ainda valores para `margin`, `padding`, `border` e `content`.

## MARGIN

A margem, como indicado do lado direito da figura 13, fez com que um espaço fosse criado entre primeira, a segunda e a última `div`. Também criou um espaçamento entre a box exemplo e a lateral esquerda da página. As margens de um elemento podem ser controladas com as propriedades CSS `margin-top`, `margin-bottom`, `margin-right` e `margin-left` – além do atalho `margin`, como visto no código de exemplo.

## BORDER

A borda delimita a área do elemento - altura e largura. Além disso, permite que uma cor diferente seja definida para essas extensões do elemento. É controlada pela propriedade CSS `border` e derivadas – com as quais a largura, a cor e o tipo de borda podem ser definidos.

O tamanho declarado para a borda é somado ao tamanho declarado para o elemento, compondo assim o seu tamanho final.

## PADDING

Para entender a função do `padding`, repare, na figura anterior, que os textos da primeira e da última `<div>` estão colados no topo e na lateral esquerda. Entretanto, na `div` do meio, há um espaçamento em relação ao topo e à lateral esquerda. Esse espaço de preenchimento equivale ao `padding`. Assim como a margem, suas dimensões são a altura e largura.

Atente-se para a seguinte diferença: `margin` diz respeito ao espaço entre elementos; já o `padding` refere-se ao conteúdo interno do próprio elemento, além de fazer parte dele. Na prática, isso significa que a `div #box_exemplo`, cuja largura foi declarada como 400px e altura como 200px tem, na verdade, 500px de altura e 300px de largura. Ou seja, o `padding` aumentou suas dimensões:  $\text{width} + \text{padding-right} + \text{padding-left} \Rightarrow 400\text{px} + 50\text{px} + 50\text{px} = 500\text{px}$   $\text{height} + \text{padding-top} + \text{padding-bottom} \Rightarrow 200\text{px} + 50\text{px} + 50\text{px} = 300\text{px}$  Para controlar o preenchimento de um elemento, são utilizadas as propriedades CSS `padding-top`, `padding-bottom`, `padding-right` e `padding-left`, além do atalho `padding`.

## CONTENT

Essa é a área interna do elemento, ocupada pelo seu conteúdo. Suas dimensões são altura e largura. Além disso, sua cor de fundo (`background`), a cor da fonte (`color`) de seu conteúdo, sua

largura (width, min-width, max-width) e altura (height, min-height, max-height) podem ser estilizadas com CSS.

# PSEUDOCLASSES E PSEUDO-ELEMENTOS

Uma declaração CSS é composta pelo elemento que se deseja estilizar, pela propriedade a ser estilizada e pelo valor a ser atribuído. Além disso, vimos que o elemento pode ser definido de maneira ampla (utilizando-se o nome da tag), específica (pelo seu identificador único) e seletiva (com a utilização de classes).

Há ainda a herança, onde um elemento filho pode herdar as propriedades de um elemento pai. Todos esses modos de definir estilo são bastante abrangentes. Entretanto, há ainda algumas formas especiais e muito úteis para se aplicar estilos: as pseudoclasses e os pseudo-elementos. Veremos a seguir as suas definições e como utilizá-las.

## PSEUDOCLASSES

As pseudoclasses são utilizadas para definir um estado especial de um elemento. Por exemplo: podemos mudar o estilo de um elemento ao passarmos o mouse sobre ele (evento mouseover). Esse novo estilo é temporário, ou seja, não corresponde ao seu estado natural. Também podemos mudar o estilo de um link que foi clicado, alterando sua cor ou alguma outra propriedade.

A sintaxe para declaração da pseudoclasse é composta pela palavra-chave correspondente ao nome da pseudoclasse precedido pelo sinal de dois pontos. Veja o exemplo a seguir:

**DIV:HOVER{BACKGROUND-COLOR:#000000;}**

Pseudoclasse	Como declarar	Para que serve

:active	a:active	Selecionar todos os links ativos.
:checked	input:checked	Selecionar todos os campos input checados.
:first-child	li:first-child	Selecionar todo primeiro item de lista.
:last-child	li:last-child	Selecionar todo último item de lista.
:hover	div:hover	Selecionar todas as divs no evento mouseover.

**Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Tabela 5: Lista básica de pseudoclasses

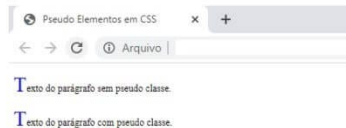
## PSEUDO-ELEMENTOS

Os pseudo-elementos são palavras-chave que, adicionadas/relacionadas a um seletor, permitem que uma parte específica dele seja estilizada. A imagem a seguir mostra duas declarações CSS, uma sem e outra com o uso de pseudo-elemento. Em ambas, é definido que a primeira letra de texto em um parágrafo tenha tamanho e cor diferentes do restante do texto nele contido.

```

1 <!doctype html>
2 <html lang="pt-BR">
3 <head>
4 <meta charset="utf-8">
5 <title>Box Model em CSS</title>
6 <style type="text/css">
7   p#sem_pseudo_classe{
8     font-size:12px;
9     color:#000000;
10  }
11
12   p#sem_pseudo_classe span.primeira_letra_fonte_maior{
13     font-size:26px;
14     color:#0000ff;
15  }
16
17   p#com_pseudo_classe{
18     font-size:12px;
19     color:#000000;
20  }
21
22   p#com_pseudo_classe::first-letter{
23     font-size:26px;
24     color:#0000ff;
25  }
26 </style>
27 </head>
28 <body>
29 <p id="sem_pseudo_classe">
30   <span class="primeira_letra_fonte_maior">T</span>exto do parágrafo sem pseudo classe.
31 </p>
32
33 <p id="com_pseudo_classe">
34   Texto do parágrafo com pseudo classe.
35 </p>
36 </body>
37 </html>

```



Fonte: Elaborado pelo autor no Notepad++.

 Figura 14: Exemplo de utilização de pseudo-elementos.

Ao analisar, codificar e testar o código acima, você perceberá que, no primeiro parágrafo, foi necessário utilizar um elemento a mais, a tag `<span>`, ao redor da primeira letra do texto para poder estilizá-la. Já no segundo parágrafo, o mesmo estilo foi alcançado apenas com o uso do pseudo-elemento `first-letter`. A utilização do pseudo-elemento diminuir a quantidade de código, tornando sua compreensão mais clara.

Cabe, ainda, destacar um outro ponto do exemplo relacionado à sintaxe dos pseudo-elementos: neles são usados dois pontos duplos (ou dobrados) para a declaração. Isso é proposital, para diferenciá-los das pseudoclasses.

Pseudo-elemento	Exemplo	Para que serve
<code>::after</code>	<code>img::after</code>	Inserir conteúdo após o elemento indicado.
<code>::before</code>	<code>h1::before</code>	Inserir conteúdo antes do elemento indicado.
<code>::first-letter</code>	<code>p::first-letter</code>	Selecionar a primeira letra do elemento indicado.

::first-line	p::first-line	Selecionar a primeira linha do elemento indicado.
::selection	p::selection	Seleciona a porção de um elemento que é selecionado pelo usuário.

**Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Tabela 6: Principais pseudo-elementos.

## POSICIONAMENTO

Para tratarmos de posicionamento em CSS, precisaremos recorrer a alguns conceitos já vistos – sobretudo os relacionados ao Box Model, além de relembrarmos que os elementos HTML possuem padrões de estilo e comportamento naturais. Alguns desses padrões diferem um pouco de um navegador para outro. Além disso, eles podem ser modificados por CSS, onde as tais diferenças são resolvidas e, de fato, um padrão de comportamento pode ser definido.

## LAYOUT EM COLUNAS E GRID LAYOUT

Esses dois conceitos são importantes quando tratamos da estrutura visual de páginas HTML. Em uma definição simplista, ambos tratam de como os elementos, boxes, podem ser posicionados e organizados em uma página. Veja na Figura 15, logo a seguir, a estrutura em colunas de uma página.

Em termos de HTML, apenas utilizando boxes (header, footer, section, aside, nav, div etc.), os elementos ficariam empilhados uns sobre os outros. Para aplicar o layout visto na Figura 15 e posicionar os elementos na página, precisaremos utilizar CSS.



## A PROPRIEDADE *POSITION*

A propriedade CSS responsável pelo posicionamento é a “position”. Seus valores possíveis são: static, relative, fixed, absolute e sticky. Além disso, as propriedades top, bottom, right e left são usadas em conjunto, a fim de definir os valores das respectivas distâncias e, consequentemente, do posicionamento. Tais propriedades, inclusive, só podem ser usadas quando for definido um valor para position.

### POSITION STATIC

Essa é a posição padrão dos elementos. Desse modo, elementos definidos como static ou sem a propriedade position são posicionados naturalmente, de acordo com o fluxo normal da página, não assumindo nenhuma localização especial. Inclusive, as propriedades top, bottom, right e left não são refletidas em elementos estáticos.

### POSITION RELATIVE

A definição da propriedade “position:relative;” para um elemento faz com que ele seja posicionado de modo relativo à sua posição normal. Com isso, ao definirmos valores para as propriedades top, bottom, right e left, ajustamos a sua posição em relação à sua posição natural.



Fonte: Elaborado pelo autor.

📷 Figura 15: Layout CSS em colunas.

## POSITION FIXED

O valor *fixed* é utilizado quando desejamos definir uma posição fixa para um elemento na página. Com isso, independente do scroll, de rolarmos a página para cima ou para baixo, o elemento sempre permanecerá no mesmo local. As propriedades *top*, *bottom*, *right* e *left* devem ser usadas para definir o lugar onde o elemento será fixado. Uma curiosidade sobre esse elemento é que ele é posicionado em relação à *viewport*/janela do navegador. Com isso, ele “flutuará” sobre os demais conteúdos da página, ficando fixo onde foi colocado e não ocupando, assim, a posição original onde foi declarado no HTML.

## POSITION ABSOLUTE

A posição “absolute” faz com que um elemento seja posicionado em relação à localização do seu elemento ancestral mais próximo – o qual também deverá estar posicionado (ou seja, não poderá ser *static*).

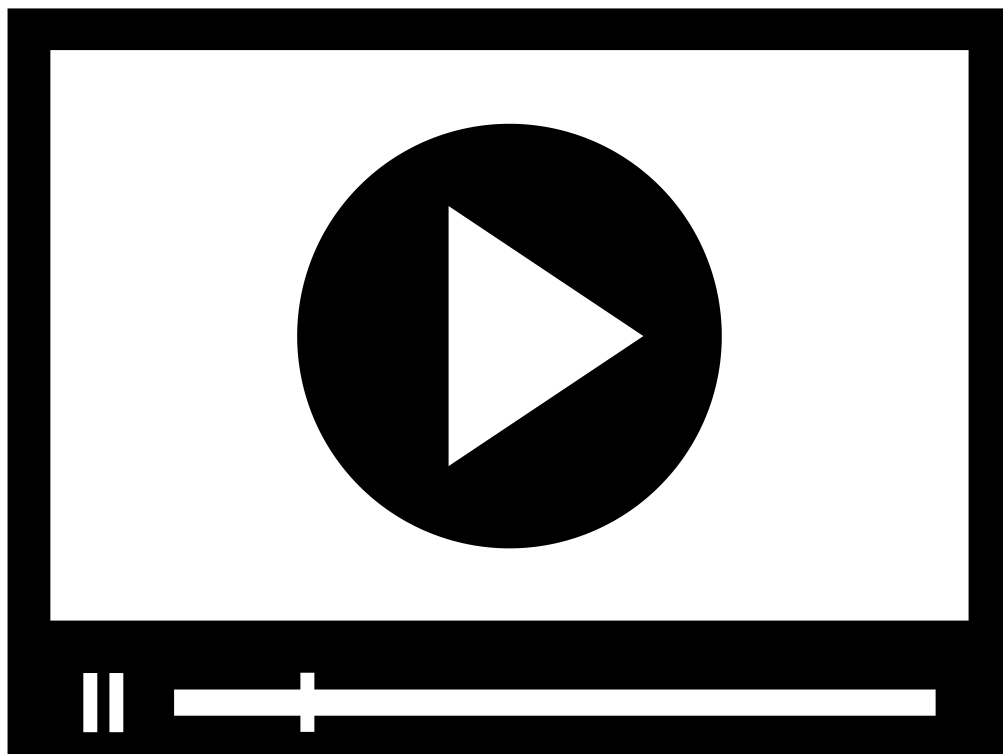
Quando o elemento definido com *absolute* for o primeiro elemento da página, ele então será posicionado em relação ao <body>. Com isso, tal elemento acompanhará a rolagem da página.

## POSITION STICKY

Esse valor para a propriedade *position* faz com que um elemento seja posicionado com base na posição de rolagem da página (scroll). Com isso, seu comportamento varia entre o relativo e o fixado, dependendo da posição do scroll.

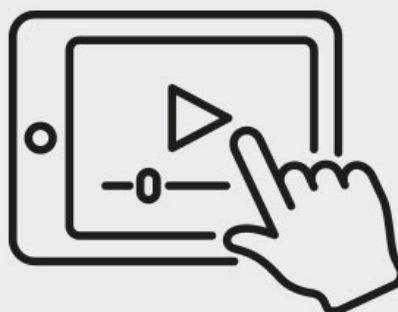
Tal propriedade é mais recente, em termos de especificação, assim não possui suporte em todas as versões dos navegadores. É usada, normalmente, quando queremos criar um efeito de sobreposição de conteúdo. Na prática, o elemento é visualizado ao abrirmos uma página. Ao rolarmos para baixo, ele se mantém fixo, com os demais conteúdos passando sob ele.





## RESULTADO DOS COMPONENTES DO BOX MODEL NO NAVEGADOR

Para assistir a um vídeo  
sobre o assunto, acesse a  
versão online deste conteúdo.



## VERIFICANDO O APRENDIZADO

**1. EM RELAÇÃO ÀS PROPRIEDADES E DIMENSÕES DO BOX MODEL REPRESENTADO PELO ELEMENTO < DIV >, CUJOS ESTILOS SÃO DEFINIDOS ABAIXO, ASSINALE A AFIRMATIVA CORRETA.**

```
DIV{  
WIDTH:500PX!IMPORTANT;  
BORDER: 5PX SOLID BLACK;  
PADDING-TOP: 10PX;  
PADDING-RIGHT:10PX;  
PADDING-BOTTOM: 5PX;  
MARGIN-LEFT:50PX;  
}
```

- A) A largura final da div será de 500px.**
- B) A largura final da div será de 520px.**
- C) A largura final da div será de 510px.**
- D) A largura final da div será de 570px.**

**2. NO FRAGMENTO DE CÓDIGO ABAIXO, A PROPRIEDADE POSITION COM O VALOR RELATIVE É DEFINIDA PARA O ELEMENTO < P >. CONSIDERANDO O CÓDIGO HTML E CSS, ASSINALE A AFIRMATIVA CORRETA.**

```
...  
< BODY >  
< DIV >  
< P > TEXTO < / P >  
< / DIV >  
< / BODY >  
...
```

---

```
P{  
POSITION:RELATIVE;  
}
```

- A) A tag < p > será posicionada de forma relativa em relação ao seu elemento ancestral, ou seja, em relação à < div >.**

**B)** A tag < p > será posicionada em função da tag < body >, uma vez que não foi declarada uma propriedade position para a < div >.

**C)** A tag < p > será posicionada da mesma forma como se nenhuma propriedade de posicionamento lhe fosse atribuída.

**D)** Para assumir a posição relativa, a tag < p > precisaria estar localizada fora da < div > ou de qualquer outro elemento pai.

---

## GABARITO

**1. Em relação às propriedades e dimensões do Box Model representado pelo elemento < div >, cujos estilos são definidos abaixo, assinale a afirmativa correta.**

```
div{  
width:500px!important;  
border: 5px solid black;  
padding-top: 10px;  
padding-right:10px;  
padding-bottom: 5px;  
margin-left:50px;  
}
```

A alternativa **"B "** está correta.

Como visto, as dimensões de largura e altura são alteradas de acordo com a borda e o padding definidos. No exemplo da questão, temos: 500px + 5px (borda da direita) + 5px (borda da esquerda) + 10px (padding da direita) = 520px.

**2. No fragmento de código abaixo, a propriedade position com o valor relative é definida para o elemento < p >. Considerando o código HTML e CSS, assinale a afirmativa correta.**

```
...  
< body >  
< div >  
< p > Texto < / p >  
< / div >
```

< / body >

...

```
p{  
position:relative;  
}
```

A alternativa "C " está correta.

As propriedades de posicionamento precisam ser utilizadas em conjunto com as propriedades top, bottom, right e left – e seus respectivos valores. Do contrário, nenhuma mudança será aplicada ao seu posicionamento. No código acima, a declaração CSS será ignorada pelo navegador.

## MÓDULO 4

---

🕒 Reconhecer Frameworks CSS

## FRAMEWORKS CSS

Como visto nos módulos anteriores, a CSS é uma tecnologia poderosa, flexível e, muitas vezes, complexa. São várias propriedades e muitos valores possíveis. Inúmeras combinações que podem, inclusive, sobrepor umas às outras.

A marcação HTML tem um comportamento natural em relação aos elementos, além de pequenas variações de um navegador para outro. Por outro lado, há bastante similaridade em relação aos layouts de diversos sites. Os de e-commerce, por exemplo, costumam ter um layout bem parecido para facilitar a experiência do usuário ao trafegar entre um e outro. Nesses casos, é comum ser utilizado o ditado de que não é necessário reinventar a roda. Esse ditado, inclusive, é um bom ponto de partida para falarmos sobre Frameworks CSS, a começar pela sua definição.

Frameworks podem ser definidos como um conjunto de componentes reutilizáveis que permitem a otimização do processo de programação.

Nesse contexto, a maioria dos Frameworks CSS mantêm similaridades entre si, além de prós e contras específicos, que vão desde a facilidade de aprendizagem, ao suporte e à documentação disponíveis, entre outros fatores.

Logo, a escolha de um Framework pode se dar por fatores objetivos, relacionados às suas características ou aos requisitos específicos de um determinado projeto, ou mesmo por fatores subjetivos, como gosto pessoal. Ao decidir utilizar um Framework, é fundamental ter em mente o quanto ele poderá auxiliá-lo em seu trabalho.

Para isso, é importante conhecer suas principais características, além das vantagens e desvantagens de cada opção. A seguir, três dos principais Frameworks CSS existentes serão apresentados: Bootstrap, Foundation e Semantic UI.



## BOOTSTRAP

Ele foi desenvolvido pela equipe do Twitter em 2011 e, posteriormente, passou a ser mantido de modo independente. Sua licença é *open source* e, atualmente, é o framework CSS mais popular.

Trata-se de um framework responsivo, baseado na premissa *mobile-first* – onde o foco inicial são os dispositivos móveis e, em seguida, os desktops. Possui componentes prontos para uso (*ready-to-use*) desenvolvidos em HTML, CSS e Javascript.

## SISTEMA DE GRID

O Grid Layout é um sistema de layout generalizado. Com ênfase em linhas e colunas, pode parecer um retorno ao layout da tabela, mas há muito mais no layout da grade do que no layout da tabela (MEYER, 2017).

Uma das principais características dos Frameworks CSS é o seu Sistema de Grids. Enquanto a Grid é um elemento de design, uma ferramenta que serve para ordenar elementos visuais, o Sistema de Grid em um Framework consiste em uma série de *containers*, linhas e colunas que servem para arranjar e alinhar conteúdo. Embora compartilhem da mesma fundamentação teórica, há pequenas diferenças de implementação entre os Frameworks.

A grid do Bootstrap, por exemplo, possui 12 colunas e 5 *breakpoints* responsivos, que são pontos de quebra nos quais o layout será ajustado para atender a diferentes resoluções de tela. Conforme visto na figura abaixo, são eles: extra small, small, medium, large e extra large.

Na prática, esse sistema deve ser corretamente utilizado para que todos os elementos da página sejam alinhados e visualizados em diferentes tamanhos de tela.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				

<b>Column ordering</b>	Yes

**Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Figura 16 – Sistema de Grid do Bootstrap. Fonte: Bootstrap Grid System, 2020.

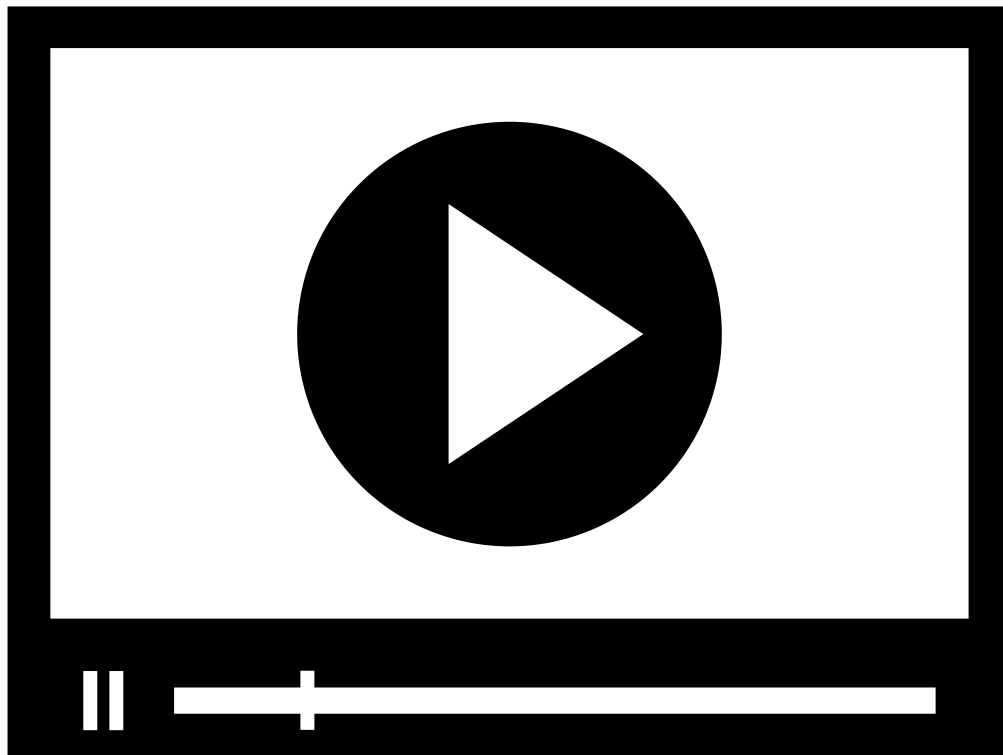
## COMO UTILIZAR O BOOTSTRAP

Para utilizar o Bootstrap, é necessário incluir a sua biblioteca, composta por dois arquivos: um CSS e outro Javascript. Essa instalação é simples e pode ser feita pela inclusão dos respectivos arquivos diretamente na HTML.

Outra forma de instalação é através de ferramentas gerenciadoras de pacotes, como **npm** ou **gem**. Em termos de dependência, para a utilização completa de todas as suas funcionalidades, é necessário ainda incluir outras bibliotecas Javascript, a JQuery e a Popper.

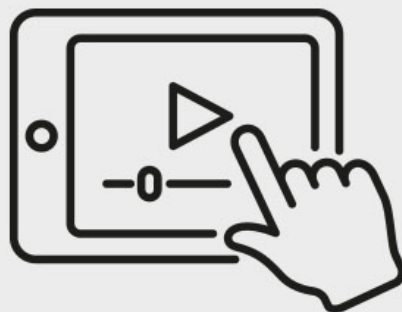
Por último, é importante ter cuidado com as versões do Framework em termos de compatibilidade, tanto em relação a bibliotecas de terceiros às JS citadas acima, quanto em relação a funcionalidades em desuso/depreciadas.

O Bootstrap possui inúmeras classes pré-definidas para as mais diversas necessidades. Para utilizá-las, é preciso combiná-las com uma marcação HTML pré-definida, conforme documentação oficial. Por exemplo: há uma classe que pode ser aplicada em tabelas para criar o efeito zebra (alternância de cores entre as linhas da tabela), a “.table-striped”. Para usar essa classe, basta incluir seu nome no atributo “class” de uma tabela.



## BOOTSTRAP

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



## FOUNDATION

O segundo Framework a ser abordado é o Foundation, criado em 2011 e que está entre os mais conhecidos e utilizados. É um framework responsivo, baseado na abordagem *mobile-first*. Sua principal característica é fazer uso, nativo, do pré-processador de CSS chamado SASS.



# SISTEMA DE GRID

O Sistema de Grid do Foundation também é composto por 12 colunas. Nas versões mais recentes, o sistema básico de grid foi substituído por um novo sistema, o XY Grid. Em tal sistema, novas funcionalidades foram adicionadas, como alinhamento vertical e horizontal, dimensionamento automático e grid vertical completa.

## COMO UTILIZAR O FOUNDATION

A forma de utilização do Foundation é semelhante à do Bootstrap: é preciso incluir um arquivo CSS e outro Javascript ou então utilizar um gerenciador de pacotes. Além disso, é recomendado também incluir a Biblioteca jQuery.

A respeito da compatibilidade, aplica-se o que foi dito anteriormente: algumas funcionalidades são descontinuadas entre uma versão ou outra. Logo, é preciso tomar cuidado para que nada deixe de funcionar ao atualizar versões.

Em termos de recursos extras, destaca-se nesse framework a existência de recursos específicos para a criação de HTML responsivo para e-mail. Trata-se do Foundation For Emails.

## SEMANTIC UI

Mais recente entre os 3 Frameworks aqui apresentados, o Semantic UI se destaca por utilizar, nativamente, um pré-processador CSS, o LESS, e a biblioteca Javascript JQuery. Também é um Framework *open source*.

Uma importante particularidade desse Framework é que suas classes utilizam sintaxe de linguagem natural, como substantivos, por exemplo, para vincular conceitos de maneira mais intuitiva.

## COMO UTILIZAR O SEMANTIC UI

A sua inclusão é semelhante às dos demais Frameworks vistos anteriormente, ou seja, por meio de arquivos CSS e JS, além da Biblioteca jQuery, ou via gerenciadores de pacotes.

## OUTROS FRAMEWORKS

Além dos 3 frameworks vistos anteriormente, há vários outros disponíveis. Alguns dos mais conhecidos são:



**PURE** Considerado o framework mais leve, foi desenvolvido pela Yahoo;

**MATERIALIZE CSS** Baseado no Material Design, ambos criados pelo Google e utilizados em seus produtos; **BULMA** Framework baseado unicamente em CSS, não faz uso de Javascript; **SKELETON** Framework minimalista. Possui apenas 400 linhas de código.

Em suma, como visto em suas definições e em seus exemplos de utilização, os Frameworks têm a mesma finalidade, servindo para as mesmas funções. Em outras palavras, tudo que é possível criar com um Framework também é possível com outro.

Por outro lado, há prós e contras em cada um deles, como o tamanho do Framework e impactos no carregamento da página, facilidade ou dificuldade de aprendizagem e simplicidade de

sintaxes. Há vários comparativos na internet que aprofundam essa discussão, elencando os pontos fortes e fracos de cada Framework.

Na prática, a recomendação é: sempre que possível, utilize um Framework. Escolha o que melhor se adequar à sua necessidade. Na dúvida por onde começar, escolha o mais utilizado – a final de contas, ele não deve ser o mais usado à toa.

## VANTAGENS E DESVANTAGENS DA UTILIZAÇÃO DE FRAMEWORKS

Na prática, não existe uma recomendação definitiva sobre usar ou não um Framework em um projeto Web. Se, por um lado, a utilização de Frameworks ajuda a otimizar o tempo de desenvolvimento, por outro lado, tira um pouco do controle do programador sobre o que está sendo feito, uma vez que não é recomendado alterar o comportamento padrão dos códigos fornecidos pelos Frameworks. A lista abaixo traz alguns prós e contras citados mais comumente:

Vantagens	Desvantagens
Padronização do código, muito válido, sobretudo, quando se trabalha em equipe;	Tamanho / peso do Framework, que pode impactar no carregamento da página;
Economia de tempo, uma vez que não é preciso criar todo o código CSS do zero;	Restrições de design – lembre-se de que o framework possui um layout padrão, baseado em grids. Isso pode acabar limitando a imaginação no momento de criação do design ou impactando no tempo de desenvolvimento para que seja possível encaixar o layout criado no padrão estabelecido pelo Framework;

Seguimento de padrões, já que os Frameworks estão sempre antenados às especificações e recomendações oficiais;	Curva de aprendizado – aprender a utilizar adequadamente um Framework pode levar algum tempo;
Compatibilidade entre navegadores.	Controle sobre o código – sendo obrigado a utilizar a estrutura definida pelo Framework, acabamos por perder o controle total sobre o código. Além disso, utilizar Frameworks sem uma boa base teórica sobre CSS pode acabar limitando o seu entendimento e aprendizado.

**Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

## VERIFICANDO O APRENDIZADO

### 1. EM RELAÇÃO À UTILIZAÇÃO DE FRAMEWORKS, ASSINALE A AFIRMATIVA INCORRETA:

- A)** Qualquer componente ou estilo disponibilizados pelos Frameworks podem ser produzidos apenas com código CSS e Javascript, ou seja, sem a utilização de Frameworks.
- B)** Os Frameworks são um importante recurso que auxiliam no desenvolvimento, diminuindo o tempo, padronizando o código e garantindo uma maior compatibilidade entre navegadores e dispositivos.
- C)** Para um melhor resultado é importante utilizar vários Frameworks em um mesmo projeto. Com isso, é possível aproveitar o que cada um oferece de melhor.

**D)** Não há um melhor ou um pior Framework. Cada um oferece vantagens e desvantagens, prós e contras. Inclusive, alguns podem ser a melhor opção para um determinado projeto e para outro não.

## **2. DENTRE AS OPÇÕES ABAIXO, ASSINALE A QUE NÃO REPRESENTA UMA VANTAGEM EM SE UTILIZAR FRAMEWORKS CSS.**

- A)** Flexibilidade e Adaptabilidade
- B)** Possibilidade de aprendizagem
- C)** Auxílio em tarefas repetitivas
- D)** Colaboração no trabalho em grupo

---

## **GABARITO**

### **1. Em relação à utilização de Frameworks, assinale a afirmativa incorreta:**

A alternativa **"C "** está correta.

A utilização de vários Frameworks CSS em um mesmo projeto pode causar inúmeros problemas, por exemplo conflitos de estilos, uma vez que alguns compartilham entre si os mesmos nomes de seletores. Logo, é imprescindível utilizar apenas um Framework por projeto.

### **2. Dentre as opções abaixo, assinale a que não representa uma vantagem em se utilizar Frameworks CSS.**

A alternativa **"A "** está correta.

O sistema de Grids dos Frameworks, embora bastante útil, acaba fazendo com que, em muitas situações, seja necessário adaptar o layout do site ao Framework, e não o contrário.

## **CONCLUSÃO**

# CONSIDERAÇÕES FINAIS

Nesse tema, foi apresentada a CSS, linguagem de estilo responsável pela camada visual em uma página Web e pela consequente separação entre apresentação e conteúdo. Ao longo dos tópicos, foram apresentados a sua definição, sua sintaxe, seus elementos, suas propriedades e formas de integração com HTML. Além disso, alguns aspectos foram abordados de maneira mais aprofundada, como as propriedades relacionadas a textos e às fontes, às pseudoclasses, aos pseudo-elementos e aos conceitos de Box Model e de posicionamento. Ao final, um importante recurso foi descrito – os Frameworks CSS. Toda a abordagem teórica foi permeada por exemplos e alguns exercícios a fim de estimular a aplicação prática e uma melhor assimilação dos conteúdos apresentados.

Para ouvir um *podcast* sobre o assunto, acesse a versão online deste conteúdo.



## REFERÊNCIAS

Meyer, E., Weyl, E. **CSS: The Definitive Guide**. O'Reilly Media, Inc., 2017.

---

## EXPLORE+

Para aprofundar seus conhecimentos sobre o Sistema de Grids, é recomendada a leitura do Livro *CSS: The Definitive Guide*, de Eric Meyer e Estelle Weyl. Indicamos também a página da w3Schools, que disponibiliza inúmeros tutoriais sobre os mais diversos assuntos relacionados ao desenvolvimento web. Busque por:

CSS Pseudoclasses

Google Fonts

CSS RGB Color

CSS HSL Color.

Indicamos também uma galeria de web fontes da Google com suporte em todos os navegadores chamada de Google Fonts.

A Mozilla disponibiliza diversos tutoriais e artigos de suporte ao desenvolvimento web.

Sugerimos a leitura de:

Fundamental text and font styling.

CSS Media Queries.

Se quiser praticar a elaboração de códigos e visualizar a renderização no navegador, indicamos dois recursos online O Codepen e o JSFiddle.

---

## CONTEUDISTA

Alexandre de Oliveira Paixão

 **CURRÍCULO LATTES**