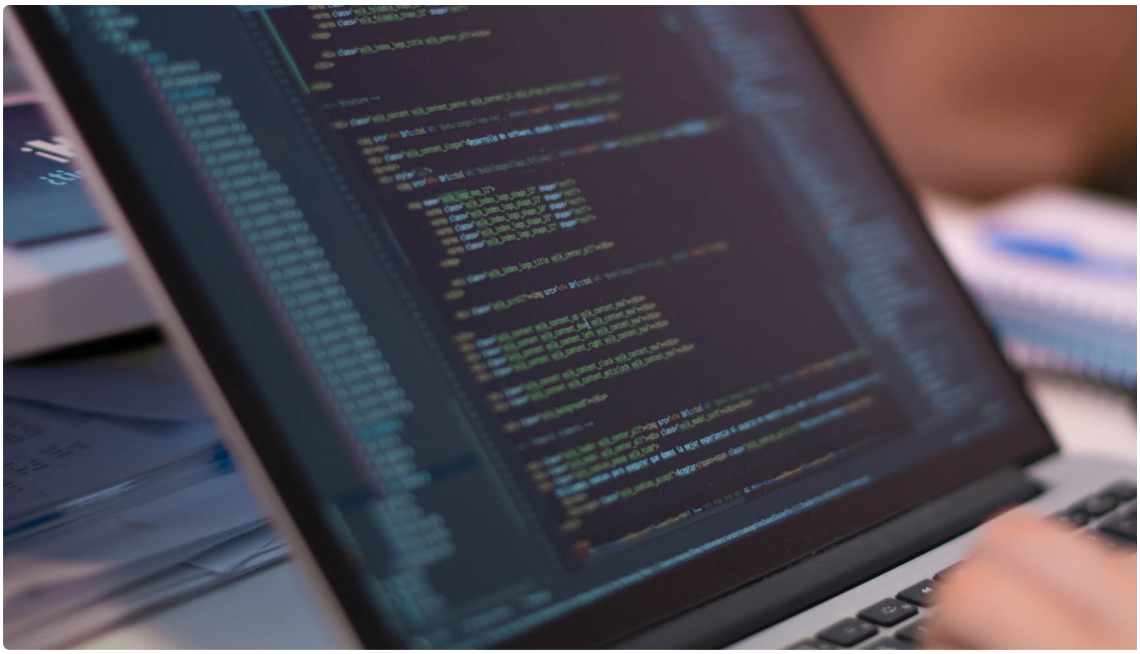


Linguagem de marcação e estilos - CSS

Prof. Alexandre de Oliveira Paixão

Prof. Kleber de Aguiar

Descrição

Conceito de folhas de estilo/CSS, recursos disponíveis, conceito de box model em uma página web. Aplicação de CSS a boxes em uma página web, conceito de seletores CSS e de pseudoclasses e pseudoelementos. Aplicação de estilos CSS utilizando pseudoclasses e pseudoelementos. 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

Fundamentos da CSS

Identificar os fundamentos da CSS.

Módulo 2

CSS3

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

Módulo 3

Conceitos avançados de CSS

Identificar os conceitos de box model, pseudoclasses, pseudoelementos e posicionamento.

Módulo 4

Frameworks CSS

Reconhecer frameworks CSS.

Introdução

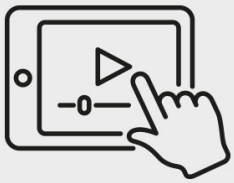
A CSS, ou folhas de Estilo em Cascata (*Cascading Style Sheets*), é uma linguagem de estilo que fornece total controle sobre a

apresentação de um documento escrito em HTML.

No ambiente web, a HTML é a linguagem responsável pela estrutura do conteúdo de uma página. Embora também seja capaz 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 o posicionamento dos elementos, as cores, tipos e tamanhos de fontes, e muito mais.

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



1 - Fundamentos da CSS

Ao final deste módulo, você será capaz de identificar os fundamentos da CSS.

Visão geral

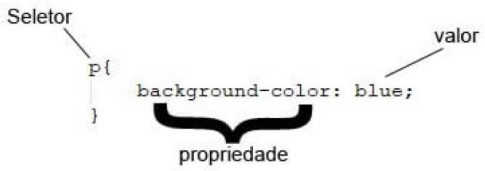
Como a CSS funciona?

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. Por meio dela, é possível, por exemplo, alterar a forma e o posicionamento dos elementos, as cores, os 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.



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 com relação 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 atribuídas à tag < p > e o resultado dessas declarações no navegador.

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

Texto do paragráfo estilizado com CSS

Demonstração da aplicação de estilos.

Sintaxe da CSS

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

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. Existem 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 seletor de identificação é definido com o atributo id.

Atenção!

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, ajudem 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. Em seguida, 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 é representado por um “.”, sendo seguidos de seus nomes. Além disso, foram apresentadas duas formas de sintaxe que produzirão o mesmo efeito. A diferença entre ambas é que, na segunda, o nome da tag à qual a identificação ou classe foi atribuída precede o respectivo sinal.

Fragmentos HTML

```
...
<h1 class="texto_vermelho">Título Principal</h1>
<p id="texto_apresentacao">
    Texto do paragráfo com atributo de identificação.
</p>
<h2>Primeiro Subtítulo</h2>
<p class="texto_descricao texto_vermelho">
    Texto do paragráfo com atributo de classe.
</p>
<h2>Segundo Subtítulo</h2>
<p class="texto_descricao">
    Texto do paragráfo 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>
```

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 verifique 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 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.

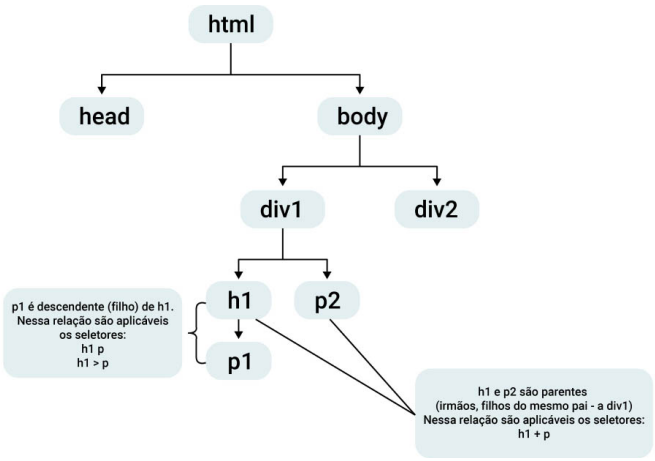
Seletores baseados em relacionamento

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).

Tabela: Principais seletores baseados em relacionamento.
Alexandre de Oliveira Paixão

Para uma melhor compreensão quanto à descendência e ao parentesco mencionados, 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.



Relação entre elementos em uma página HTML.

Propriedades CSS

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	Estabiliza 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, Righ e Left	Controlam o posicionamento, relativo ou absoluto, dos elementos em relação a outros elementos.

Propriedade	Função
Color	Estila a cor do conteúdo textual de um elemento.
Font-family, Font-size, Font-weight etc.	Série de propriedades usada 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	Define a altura de um elemento.
List-style, List-style-image etc.	Propriedades usadas 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	Define a largura de um elemento.
Z-index	Define a profundidade de um elemento – usado, por exemplo, para sobreposição de

Propriedade	Função
	elementos.

Tabela: Lista de propriedades CSS mais comuns.
Alexandre de Oliveira Paixão

CSS e HTML

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?

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.

A declaração inline faz uso do atributo style procedido por declarações separadas por ponto e vírgula ";". Esse atributo pode ser usado em qualquer tag HTML.

```
<p style="background-color: blue;">Texto paragráfo</p>
```

Aplicação de estilo inline.

CSS interna

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

```
<html>
    <head>
        <style type="text/css">
            p{
                background-color: blue;
            }
        </style>
    ...
    </head>
...
</html>
```

Aplicação de estilo CSS interna.

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:

```
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="estilos.css" />
    ...
    </head>
...
</html>

<html>
    <head>
        <style>
            @import url("estilos.css");
        </style>
    ...
    </head>
...
</html>
```

Aplicação de estilo externa.

CSS em escopo

Essa forma de aplicação de estilo foi criada a partir da HTML5. Por meio dela, é possível aplicar estilos no âmbito de escopo, ou seja, específicos para as seções da página em que 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 >.

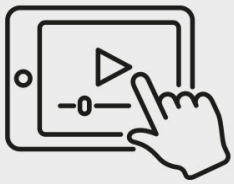
```
...
<div>

    <style type="text/css">
        /* Estes estilos serão aplicados apenas dentro da Div */
        p{
            background-color: blue;
        }
    </style>
    <p>Texto do parágrafo</p>

</div>
...
```

Aplicação de estilo a nível de escopo.

Integrando a CSS à HTML

Neste vídeo, mostraremos como integrar 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

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

```
...

div{
    color: blue;
}

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

...
```

Texto solto na Div

Texto do parágrafo que é "filho" da Div

Exemplo de herança em CSS.

O resultado do fragmento de código 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 herdar 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.

```
...

div{
    color: blue;
}

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

...
```

Texto solto na Div

Texto do parágrafo que é "filho" da Div

Exemplo de especificidade em CSS.

Perceba que o fragmento da figura exemplo de herança em CSS 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

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, ou seja, 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 esta ordem:

1

Os estilos internos e de escopo têm precedência sobre estilos em arquivos externos.

2

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

Quanto aos seletores, a **regra de precedência** segue esta 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.

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>
```

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 ocultar ou mostrar determinada declaração de acordo com o navegador, sua versão ou recursos disponíveis.

Dicas sobre as regras de precedência

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

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.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 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.

E A HTML provê basicamente duas formas para aplicação de estilos: CSS inline e CSS interna. Somente a partir da HTML5 mais duas formas foram disponibilizadas: CSS externa e CSS de escopo.

Parabéns! 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.

Questão 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; }

E p { background-color: green; }

Parabéns! A alternativa C está correta.

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

2 - CSS3

Ao final deste módulo, você será capaz de reconhecer os recursos de cores, texto, fontes e web fontes da CSS3.

Recursos de cores

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, nas quais podem ser usados os nomes das cores (seguindo as definidas pela especificação CSS) ou a notação hexadecimal. Por exemplo: 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

Essas propriedades se referem a quais elementos podemos definir cores.

Veja na tabela a seguir as principais 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.

Tabela: Lista das propriedades de cor e elementos nos quais podem ser aplicadas.
Alexandre de Oliveira Paixão

Cores

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

Recursos de textos e fontes

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. Veja a seguir algumas propriedades CSS que alteram esse comportamento padrão.

O layout do texto

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. Vamos conhecer, a seguir, 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 *container* no qual está inserido.

Tal propriedade pode assumir quatro valores: left, right, center e justify. Como os nomes indicam, essas propriedades alinham o texto à esquerda, à 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 para declarar o espaçamento entre linhas é utilizando o valor em número. Desse modo, o espaçamento será o resultado da multiplicação do valor definido pelo tamanho da fonte.

Exemplo

Line-height: 1.5; font-size: 12px; onde valor 1.5 será multiplicado pelo valor da propriedade font-size, resultando no valor de 18px de espaçamento.

Espaçamento entre letras e palavras

As propriedades letter-spacing e word-spacing permitem alterar o espaçamento entre letras e/ou palavras. 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 essa propriedade, 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 determinada fonte não esteja disponível no dispositivo cliente, a próxima fonte definida será usada, e assim sucessivamente. Caso nenhuma das fontes definidas esteja disponível 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. Essas fontes e algumas outras formam o conjunto chamado 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, além de porcentagem 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 de 100 a 900.

Existem boas práticas e cuidados a serem levados em consideração quando se trabalha com estilização de fontes usando CSS. Um desses cuidados 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 na qual 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');
}
```

Declaração de Web Fonte.

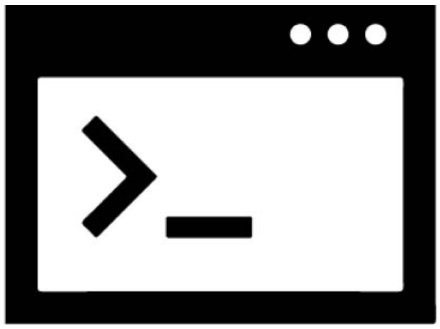
Como podemos observar, além das propriedades font-family e src, há outras que podem ser aplicadas às web fontes. Em relação à font-family, a partir do momento da sua declaração, o nome definido poderá ser utilizado para estilizar qualquer outro elemento ao longo do CSS – considere que podemos tanto utilizar 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 outros elementos na declaração dos quais ainda não falamos. São eles: as funções **local** e **format**.

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

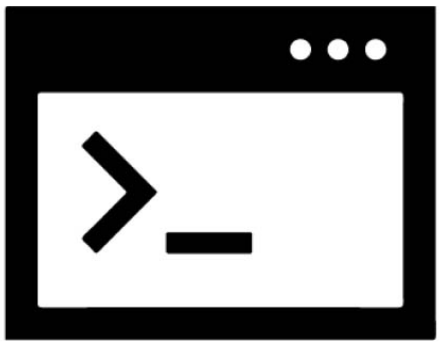
A função local

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.



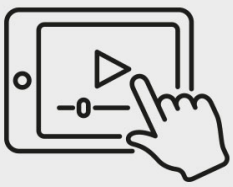
A função format

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”.



Como utilizar a regra @font-face

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



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
TTF/OTF	4.0	3.5	9.0	3.1
WOFF	5.0	3.6	9.0	5.1
WOFF2	36.0	35.0	--	--
SVG	4.0	--	--	3.2
EOT	--	--	6.0	--

Tabela: Tipos mais comuns de web fontes.
Alexandre de Oliveira Paixão

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:

Atenção!

Considere sempre utilizar diferentes fontes para oferecer uma melhor experiência aos usuários.

Abreviaturas ou atalhos

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:

```css
p{
    margin-top: 10px;
    margin-bottom: 8px;
    margin-right: 6px;
    margin-left: 4px;
}
```

```css
p{
    margin: 10px 6px 8px 4px;
}
```

```css
p{
    padding-top: 10px;
    padding-bottom: 8px;
    padding-right: 6px;
    padding-left: 4px;
}
```

```css
p{
    padding: 10px 6px 8px 4px;
}
```

```css
p{
    border-width: 2px;
    border-style: solid;
    border-color: #cccccc;
}
```

```css
p{
    border: 2px solid #cccccc;
}
```

```css
p{
    background-color: #000000;
    background-image: url(imagem.jpg);
    background-repeat: no-repeat;
    background-position: top left;
}
```

```css
p{
    background: #000000 url(imagem.jpg) no-repeat top left;
}
```

```css
p{
    font-size: 1em;
    line-height: 1.5em,;
    font-weight: bold;
    font-style: italic;
    font-family: verdana;
}
```

```css
p{
    font: 1em/1.5em bold italic verdana;
}
```

```css
ul{
    list-style: #000000;
    list-style-type: disc;
    list-style-position: outside;
    list-style-image: url(imagem.jpg);
}
```

```css
ul{
    list-style: disc outside url(imagem.jpg);
}
```

Fontes

Neste vídeo, veremos as propriedades CSS relacionadas a fontes.

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

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 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 do estilo 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.

E a CSS permite controle apenas parcial sobre os elementos de fonte, pois alguns navegadores padronizam os estilos desses elementos. Por outro lado, em relação aos elementos de texto, o controle é total e exclusivo dos desenvolvedores, independentemente do estilo padrão dos navegadores que venham a exibir a página.

Parabéns! 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.

Questão 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 por meio 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 nem web fontes, 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 este possuir ou não a fonte definida.

E

Para obter maior garantia e controle sobre o resultado final a ser exibido ao usuário, evitando a ocorrência de possíveis degradações na página, é necessário optar pela utilização de fontes genéricas em detrimento da utilização de uma lista de fontes.

Parabéns! 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.

3 - Conceitos avançados de CSS

Ao final deste módulo, você será capaz de identificar os conceitos de box model, pseudoclasses, pseudoelementos e posicionamento.

Conceitos de 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.

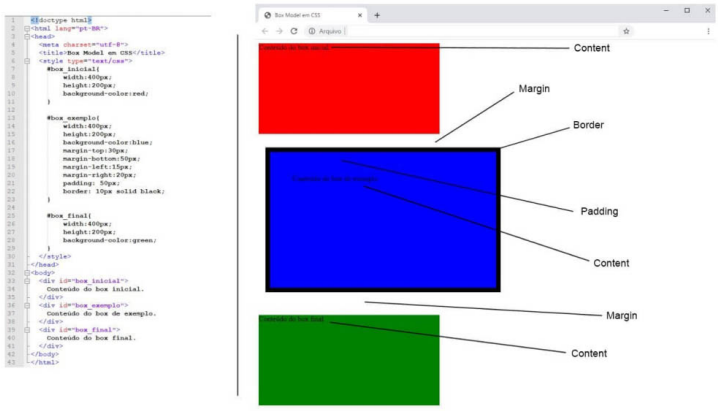
Neste módulo, avançaremos um pouco mais e percorreremos os conceitos de Box Model (modelo de caixas ou retângulos), Pseudoclasses e Pseudoelementos e Posicionamento.

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.

Nos brinquedos de bloco, 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).



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 Componentes do Box Model, 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 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: width + padding-right + padding-left => 400px + 50px + 50px = 500px height + padding-top + padding-bottom => 200px + 50px + 50px = 300px.

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.

Resultado dos componentes do box model no navegador

Neste vídeo, veremos a representação dos componentes do box model em uma página web.

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

Conceitos de pseudoclasses e pseudoelementos

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).

Um elemento filho pode, ainda, herdar as propriedades de um elemento pai. Todos esses modos de definir estilo são bastante abrangentes. Entretanto, existem algumas formas especiais e muito úteis para se aplicar estilos: as **pseudoclasses** e os **pseudoelementos**. 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:

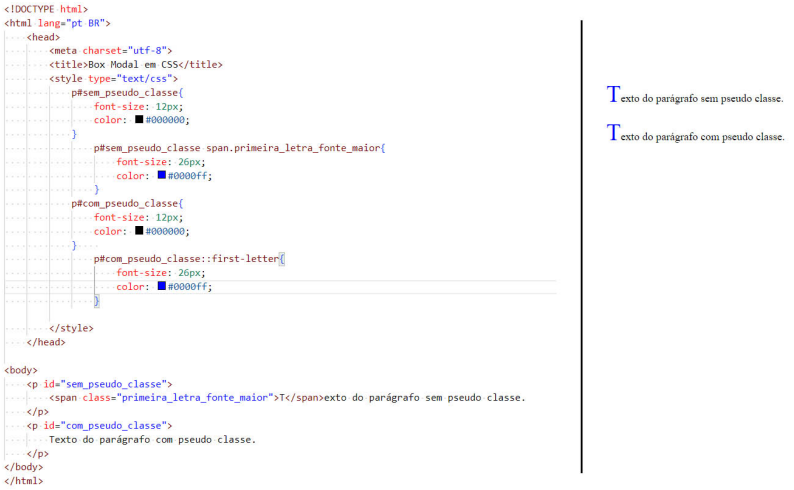
div:hover{background-color:#000000;}

Pseudoclasse	Como declarar	Para que serve
:active	a:active	Seleciona todos links ativos.
:checked	input:checked	Seleciona todos campos input checados.
:first-child	li:first-child	Seleciona todo primeiro item de lista.
:last-child	li:last-child	Seleciona todo último item de li
:hover	div:hover	Seleciona todas divs no evento mouseover.

Tabela: Lista básica de pseudoclasses
Alexandre de Oliveira Paixão

Pseudoelementos

Os pseudoelementos 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 pseudoelemento. Em ambas, é definido que a primeira letra de texto em um parágrafo tenha tamanho e cor diferentes do restante do texto.



```
<!DOCTYPE html>
<html lang="pt BR">
    <head>
        <meta charset="utf-8">
        <title>Box Modal em CSS</title>
        <style type="text/css">
            p#sem_pseudo_classe{
                font-size: 12px;
                color: #000000;
            }
                p#sem_pseudo_classe span.primeira_letra_fonte_maior{
                    font-size: 26px;
                    color: #0000ff;
                }
            p#com_pseudo_classe{
                font-size: 12px;
                color: #000000;
            }
                p#com_pseudo_classe::first-letter{
                    font-size: 26px;
                    color: #0000ff;
                }

        </style>
    </head>

<body>
    <p id="sem_pseudo_classe">
        <span class="primeira_letra_fonte_maior">T</span>exto do parágrafo sem pseudo classe.
    </p>
    <p id="com_pseudo_classe">
        Texto do parágrafo com pseudo classe.
    </p>
</body>
</html>
```

Texto do parágrafo sem pseudo classe.

Texto do parágrafo com pseudo classe.

Exemplo de utilização de pseudoelementos.

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 pseudoelemento first-letter. A utilização do pseudoelemento diminui a quantidade de código, tornando sua compreensão mais clara.

Cabe destacar outro ponto do exemplo relacionado à sintaxe dos pseudoelementos: neles são usados dois pontos duplos (ou dobrados) para a declaração. Esse uso proposital é para diferenciá-los das pseudoclasses.

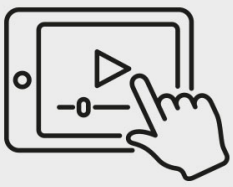
Pseudo-elemento	Exemplo	Para que serve
::after	img::after	Inserir conteúdc após o elemento indicado.
::before	h1::before	Inserir conteúdc antes do elemer

Pseudo-elemento	Exemplo	Para que serve
		indicado.
::first::letter	p::first-letter	Selecionar a primeira letra do elemento indica
::first-line	p::first-line	Selecionar a primeira linha do elemento indica
::selection	p::selection	Seleciona a porç de um elemento que é selecionad pelo usuário.

Tabela: Principais pseudoelementos
Alexandre de Oliveira Paixão

Pseudoclasses e Pseudo-Elementos

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



Posicionamento

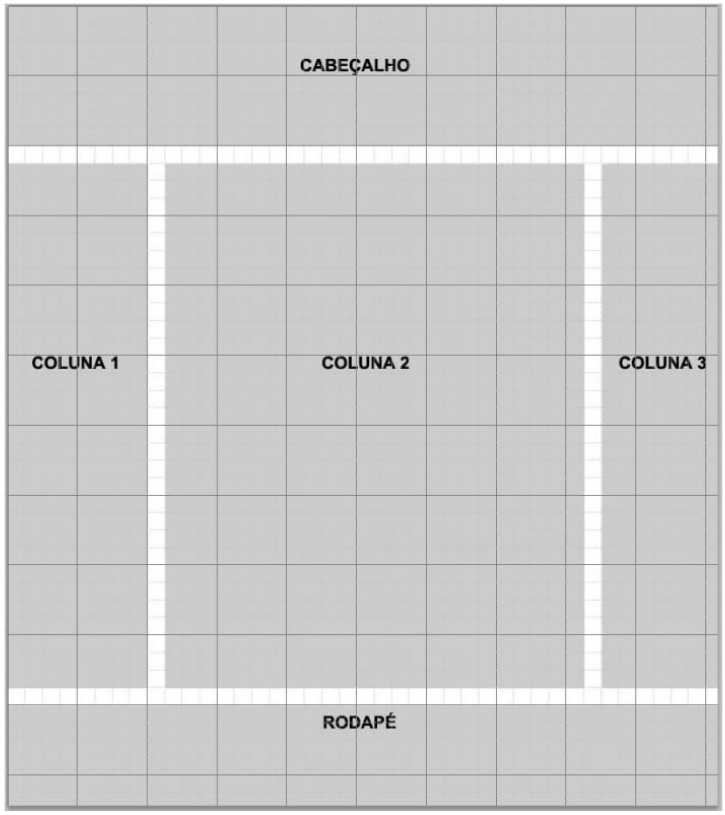
Para tratarmos de posicionamento em CSS, precisaremos recorrer a alguns conceitos já vistos – sobretudo aos relacionados ao Box Model, além de relembrarmos que os elementos HTML possuem padrões de estilo e comportamento naturais. Alguns desses padrões diferem um pouco de um navegador para outro. Além disso, podem ser modificados

por CSS, onde as tais diferenças são resolvidas e, de fato, um padrão de comportamento pode ser definido.

Conceitos de layout

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 layout CSS em colunas, logo a seguir, a estrutura em colunas de uma página.



Layout CSS em colunas.

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 layout CSS em colunas 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. A seguir, apresentaremos cada uma dessas cinco propriedades.

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.

Position fixed

O valor **fixed** é utilizado quando desejamos definir uma posição fixa para um elemento na página. Com isso, independentemente 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 no qual o elemento será fixado. Esse elemento 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 na qual 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 como 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.

A propriedade *position*

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

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 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.

E A largura final da div será de 590px.

Parabéns! 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.

Questão 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.

E Por estar localizada dentro da < div >, a tag < p > será posicionada em função da tag < body >, que é um elemento ancestral da tag < div >.

Parabéns! 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.

4 - Frameworks CSS

Ao final deste módulo, você será capaz de reconhecer Frameworks CSS.

Visão geral

Frameworks e 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 podem, inclusive, se 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 sites 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 mantém 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 as 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**.

Frameworks

Bootstrap

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* – cujo 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. Esses breakpoints são:

Extra small

Small

Medium

Large

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
Max container width	None (auto)	540px	720px	960px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-

	Extra small <576px	**Small** ≥576px	**Medium** ≥768px	**Large** ≥992px
# of columns	12			
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Column ordering	Yes			

Tabela: Sistema de grid do Bootstrap.
Alexandre de Oliveira Paixão

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 por meio 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 quanto em relação a funcionalidades em desuso/depreciadas.

Comentário

O Bootstrap possui inúmeras classes predefinidas para as mais diversas necessidades. Para utilizá-las, é preciso combiná-las com uma marcação HTML predefinida, 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.

Demonstração usando framework Boostrap

Neste vídeo, veremos sobre os conjuntos de componentes reutilizáveis: os frameworks CSS.

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 de 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.

Comentário

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 à 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, sendo os mais conhecidos os seguintes:

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 tamanho do framework e seus 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, é recomendável, sempre que possível, utilizar um framework. Escolha aquele que melhor se adequar à sua necessidade. Na dúvida, escolha o mais utilizado – afinal de contas, ele não deve ser o mais usado à toa.**

Foundation e Semantic UI

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

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 seguir, listeremos algumas **vantagens**, bem como **desvantagens**, que são citadas de maneira mais comumente.

Como **vantagens**, podemos destacar:

Padronização do código

Muito válido, sobretudo quando se trabalha em equipe.

Economia de tempo

Não é preciso criar todo o código CSS do zero.

Seguimento de padrões

Os frameworks estão sempre antenados às especificações e recomendações oficiais.

Compatibilidade entre navegadores

Funcionam em diferentes navegadores.

A seguir, listaremos algumas **desvantagens**:

Tamanho/peso do framework

Pode impactar no carregamento da página.

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.

Curva de aprendizado

Aprender a utilizar adequadamente um framework pode levar algum tempo.

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 limitar o seu entendimento e aprendizado.

Vantagens e desvantagens da utilização de Frameworks

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

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 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.

E A escolha do framework a ser utilizado é complexa, pois eles são bem distintos. Apenas uma minoria dos frameworks CSS mantém similaridades entre si.

Parabéns! 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.

Questão 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.

E Garantia de controle sobre o código.

Parabéns! 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.

Considerações finais

Este estudo nos apresentou 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 apresentadas a sua definição, sua sintaxe, seus elementos, suas propriedades e formas de integração com HTML. Alguns aspectos foram abordados de maneira mais aprofundada, como as propriedades relacionadas a textos e às fontes, as pseudoclasses, os pseudoelementos e os 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.

Podcast

Para encerrar, ouça um breve resumo sobre a linguagem de marcação e estilos CSS.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.

Explore +

Para aprofundar seus conhecimentos sobre o Sistema de Grids, recomendamos 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. Mozilla também 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 on-line: o Codepen e o JSFiddle.

Referências

Meyer, E.; WEYL, E. **CSS**: The Definitive Guide. [s.l.]: O'Reilly Media, Inc., 2017.

Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

Download material

O que você achou do conteúdo?

Relatar problema