Objetivos:

- Linguagem de marcação HTML;
- Elementos em bloco versus elementos inline;
- Estrutura de um documento HTML;
- Caracteres especiais no documento HTML;
- Comentários no HTML;
- Principais elementos do corpo de uma página HTML;
- Imagens no HTML;
- Tabelas no HTML;
- Formulários HTML;
- Linguagem de formatação CSS (Cascading Style Sheets);
- Flexbox layout.

i. Linguagem de marcação HTML

HTML (HyperText Markup Language) não é uma linguagem de programação, é uma linguagem de marcação utilizada para dizer ao navegador como estruturar a página web. Ela utiliza marcações para representar textos, imagens e outros conteúdos para exibição no navegador.

Algumas considerações:

- Uma página HTML é chamada de documento;
- Uma página é formada por marcações;
- Uma marcação é chamada de elemento HTML;
- O elemento HTML é formado pelo nome da marcação (tagname), atributos e conteúdo. No exemplo a seguir, a marcação `</p>` é o fechamento do elemento e `Boa dia` é o conteúdo do elemento:

```
<p>Bom dia</p>
```

  Algumas marcações não possuem conteúdo, pois elas são formadas apenas pela marcação de abertura. A marcação de imagem não requer marcação de fechamento e consequentemente não possui conteúdo:

```
<img src='teste.png' title='Detalhes'>
```

  Geralmente os elementos que possuem somente a marcação de abertura são usadas para inserir/incorporar algo no documento no lugar em que ele é incluído, como exemplo, a imagem é um arquivo a parte que é incorporado no documento pelo navegador.

  Algumas marcações podem ter atributos, nesse exemplo `src` e `title` são atributos do elemento imagem. Os atributos são sempre colocados na marcação de abertura.

- Um atributo deve conter:
  - Um espaço entre ele e o nome do elemento (ou o atributo anterior, caso o elemento já contenha um ou mais atributos);
  - O nome do atributo, seguido por um sinal de igual;
  - Um valor para o atributo, entre aspas simples ou duplas.

- Os atributos booleanos são aqueles que podem ter somente um valor, que é geralmente o mesmo nome do atributo. No exemplo a seguir, o atributo `disabled` é usado para desabilitar o campo de entrada do formulário, neste caso podemos omitir o valor do atributo:

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

- É obrigatório incluir a marcação de fechamento no elemento que requer fechamento. O exemplo a seguir está errado, pois o elemento de parágrafo requer marcação de fechamento:

```
<p>Bom dia
```

- Elementos podem ser aninhados, desde que o elemento filho possa ser colocado dentro do elemento pai, no exemplo a seguir o elemento `<p>` aceita o elemento `<strong>` como filho:

```
<p>Bom dia <strong>Ana</strong></p>
```

  Ao aninhar elementos é importante fechar as marcações na ordem em que elas foram abertas, o exemplo anterior estaria errado se invertêssemos a ordem de fechamento das marcações:

```
<p>Bom dia <span>Ana</p></span>
```

- O nome de um elemento (tagname) é insensível a maiúsculas e minúsculas, isto é, pode ser escrito em maiúsculas, minúsculas ou misto. Porém, constitui boa prática manter em minúsculo.

Para mais detalhes acesse https://developer.mozilla.org/pt-BR/docs/Web/HTML.

ii. Elementos em bloco versus elementos inline

Os elementos HTML podem ser categorizados “elementos em bloco” e “elementos inline”:

- Elementos em bloco formam um bloco visível na página. Eles aparecerão em uma nova linha logo após qualquer elemento que venha antes dele, e qualquer conteúdo depois de um elemento em bloco também aparecerá em uma nova linha. Elementos em bloco geralmente são elementos estruturais na página que representam, por exemplo: parágrafos, listas, menus de navegação, rodapés etc. Um elemento em bloco não deve ser aninhado dentro de um elemento inline, mas pode ser aninhado dentro de outro elemento em bloco.
- Elementos inline (na linha) são aqueles que estão contidos dentro de elementos em bloco, envolvem apenas pequenas partes do conteúdo do documento e não parágrafos inteiros ou agrupamentos de conteúdo. Um elemento inline não fará com que uma nova linha apareça no documento: os elementos inline geralmente aparecem dentro de um parágrafo de texto, por exemplo: um elemento <a> (hyperlink) ou elementos de ênfase como ou .

No exemplo a seguir, O elemento `<em>` é inline, então os três primeiros elementos ficam na mesma linha uns dos outros sem espaço entre eles. O `<p>`, por outro lado, é um elemento em bloco, então cada elemento aparece em uma nova linha, com espaço acima e abaixo de cada um (o espaçamento é devido à estilização CSS padrão que o navegador aplica aos parágrafos).

```
<em>primeiro</em><em>segundo</em><em>terceiro</em>

<p>quarto</p><p>quinto</p><p>sexto</p>
```

primeirosegundoterceiro

quarto

quinto

sexto

Para mais detalhes sobre elementos em bloco acesse https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements e para detalhes sobre elementos inline acesse https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements.

iii. Estrutura de um documento HTML

O código HTML a seguir possui os elementos mínimos para construirmos a estrutura de uma página.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Título</title>
    </head>
    <body>
        <p>Corpo da página</p>
    </body>
</html>
```

Nesse código temos:

- `<!DOCTYPE html>`: Por volta do ano de 1991/2, doctypes funcionavam como links para uma série de regras as quais uma página HTML tinha de seguir para ser considerada bem formatada, o que poderia significar a verificação automática de erros e outras coisas úteis. Ele costumava ser assim:

  ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  ```

  Atualmente, pode-se dizer que doctype não é uma marcação, e sim uma instrução ao navegador a respeito de qual versão do HTML foi utilizada no código. A marcação `<!DOCTYPE html>` indica que o documento foi codificado na versão HTML 5.
- `<html></html>`: O elemento `<html>` envolve o conteúdo da página inteira e é conhecido como o "elemento raiz" da página HTML.
- `<head></head>`: O head de um documento HTML é a parte que não é exibida no navegador quando a página é carregada. O elemento `<head>` recebe as marcações utilizadas para instruir o navegador na construção da página, tais como, codificação dos caracteres (`<meta charset="UTF-8">`), descrição da página (`<meta name="description" content="Aprendendo HTML">`) que é usada pelos mecanismos de busca, autor (`<meta name="author" content="Professor">`) e configurações sobre a área visível na tela do dispositivo (`<meta name="viewport" content="width=device-width, initial-scale=1.0">`).

  ```
  <head>
      <meta charset="UTF-8">
      <meta name="description" content="Aprendendo HTML">
  ```

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

Para mais detalhes sobre o elemento head acesse https://developer.mozilla.org/pt-BR/docs/Aprender/HTML/Introducao_ao_HTML/The_head_metadata_in_HTML.

- `<meta charset="utf-8">`: Se não informarmos a codificação do documento, o navegador usará aquela definida nas configurações do usuário, isso fará diferença nos caracteres especiais, tal como, na palavra Sebastião (SebastiÃƒÂ£o);
- <title></title>: Define o título da página, que aparecerá na guia do navegador na qual a página está carregada e é usado para descrevê-la quando o usuário salvar nos favoritos.
- <body></body>: Contém todo o conteúdo a ser mostrado aos usuários na tela do navegador, tais como, textos, imagens, vídeos, links etc.

iv. Caracteres especiais no documento HTML

Os caracteres <, >,",' e & são especiais no HTML, pois eles já são usados na sintaxe da linguagem de marcação do HTML. Então para termos eles nas nossas páginas teremos de usar as "referências de caracteres", que são códigos especiais que representamos caracteres. Cada referência de caractere começa com o "e" comercial (&), e termina com o ponto e vírgula (;). A seguir tem-se alguns exemplos:

```
<div>2 &lt; 3</div>
<div>2 &le; 3</div>
<div>3 &gt; 2</div>
<div>3 &ge; 2</div>
<div>Aspas &quot;duplas&quot;, &apos;simples&apos; e &amp;</div>
```

2 < 3
2 ≤ 3
3 > 2
3 ≥ 2
Aspas "duplas", 'simples' e &

Para outros exemplos de caracteres especiais acesse https://www.html.am/reference/html-special-characters.cfm.

v. Comentários no HTML

Os marcadores <!-- e --> são usados para marcar o bloco a ser ignorado pelo navegador, isto é, ficará invisível para o usuário:

```
<!-- <div>2 &lt; 3</div> tudo aqui será ignorado pelo navegador -->
```

vi. Principais elementos do corpo de uma página HTML

Crie um arquivo de nome ex1.html usando o Visual Studio Code e copie os exemplos a seguir para o arquivo ex1.html para testar.

1 – Elementos de texto

De uma forma simplista podemos reduzir os textos de uma página a títulos e parágrafos. O elemento <p> (parágrafo) cria um parágrafo e os elementos <h1>, <h2>, <h3>, <h4>, <h5> e <h6> são usados para criar título e subtítulos. Cada elemento representa um nível diferente de conteúdo no documento; <h1> representa o título principal, <h2> representa subtítulos, <h3> representa sub-subtítulos, e assim por diante. No exemplo a seguir usamos algumas marcações para formar um texto com 3 níveis de títulos e alguns parágrafos.

```
<!DOCTYPE html>
```

```
<html>
    <head>
        <meta charset="utf-8">
        <title>Título</title>
    </head>
    <body>
        <h1>Front-end</h1>
        <h2>Linguagem de marcação</h2>
        <p>HTML é usada para estruturar a página, todo o conteúdo de uma página precisa estar
           nos elementos HTML.</p>
        <h2>Estilos</h2>
        <p>O CSS é a linguagem usada para formatar os elemntos HTML.</p>
        <p>O código CSS pode estar em diferentes locais.</p>
        <h3>Inline</h3>
        <p>Na própria marcação como um atributo.</p>
        <h3>Incorporado</h3>
        <p>No próprio arquivo .html, mas dentro do element &lt;style&gt;.</p>
        <h3>Importado</h3>
        <p>O código CSS fica fica separado noutro arquivo .css e precisa ser incluído
           usando o element &lt;link&gt;.</p>
    </body>
</html>
```

Front-end

Linguagem de marcação

HTML é usada para estruturar a página, todo o conteúdo de uma página precisa estar nos elementos HTML.

Estilos

O CSS é a linguagem usada para formatar os elemntos HTML.

O código CSS pode estar em diferentes locais.

Inline

Na própria marcação como um atributo.

Incorporado

No próprio arquivo .html, mas dentro do element <style>.

Importado

O código CSS fica fica separado noutro arquivo .css e precisa ser incluído usando o element <link>.

Observe que os elementos `<p>` e `<h*>` já possuem uma formatação padrão de margens. O elemento `<div>` também pode ser usado para colocarmos texto, mas ele não possui formatação. Como exemplo, substitua os seguintes elementos de `<p>` para `<div>` e veja no navegador que as linhas estarão sem margem vertical.

```
<div>O CSS é a linguagem usada para formatar os elemntos HTML.</div>
<div>O código CSS pode estar em diferentes locais.</div>
```

O elemento também é usado para marcar textos, mas ele não é um elemento estrutural, pois não tem semântica. Usamos ele para embrulhar conteúdo quando desejamos aplicar algum estilo CSS ou fazer algo usando JavaScript.

2 – Listas ordenadas e não ordenadas

Os elementos <ol> (ordered list) e <ul> (unordered list) são usados para criar listas, porém eles elementos sozinhos não são capazes de criar as listas, pois uma lista é formada por itens, que são criados pelos elementos <li> (list item). A seguir temos alguns exemplos de listas.

```
<body>
    <h3>Lista sem ordem</h3>
    <ul>
        <li>Abacaxi</li>
        <li>Limão</li>
        <li>Pêra</li>
    </ul>
    <h3>Lista ordenada</h3>
    <ol>
        <li>Alface</li>
        <li>Repolho</li>
        <li>Tomate</li>
    </ol>
    <h3>Lista aninhada</h3>
    <ol>
        <li>Grãos</li>
        <li>Frutas:
            <ul>
                <li>Maça</li>
                <li>Mamão</li>
            </ul>
        <li>Verduras</li>
    </ol>
</body>
```

Lista sem ordem

- Abacaxi
- Limão
- Pêra

Lista ordenada

1. Alface
2. Repolho
3. Tomate

Lista aninhada

1. Grãos
2. Frutas:
    - Maça
    - Mamão
3. Verduras

3 – Elemento de hyperlink

O elemento hyperlink <a> é o que torna a web uma web, pois é através deles que navegamos de uma página para outra.

Um link é criado envolvendo um texto ou outros elementos HTML com a marcação <a>. No exemplo a seguir os textos G1 e UOL foram envolvidos pela marcação <a>, desta forma, a navegação de uma página para outra irá ocorrer quando o usuário clicar no texto a G1 ou UOL. Já no outro caso a navegação ocorrerá quando o usuário clicar na imagem, pois a marcação <a> envolve um elemento de imagem <img>.

```
<body>
    <a href="http://g1.globo.com">G1</a>
    <a href="https://www.uol.com.br" target='_blank' title='Clique para ir'>UOL</a>
    <a href='https://www.google.com/'>
        <img src='logo.png' alt='Google'>
    </a>
```

```
        <a>Não é um hyperlink</a>
        <a href='#' target='_blank'>Página atual</a>
    </body>
```

O elemento `<a>` precisa do atributo `href` para identificar o destino do link, isto é, a marcação a seguir não é um hyperlink.

```
    <a>Não é um hyperlink</a>
```

O valor `#` no atributo `href` é usado para fazer um link para a própria página, isto é, quando o usuário clicar no texto `Página atual` o navegador irá abrir a página atual novamente.

```
    <a href='#' target='_blank'>Página atual</a>
```

O atributo `target` com valor `_blank` indica que o link será aberto noutra aba do navegador, sendo que o padrão é `_self`, que significa abrir na aba atual.

O atributo `title` é usado para exibir um texto quando o usuário posicionar o cursor sobre o elemento.

Podemos criar um vínculo para uma parte específica do documento HTML (conhecido como um fragmento de documento) e não apenas para a página como um todo. Para fazer isso, precisamos fazer o vínculo usando o atributo `id` (identificador único de elemento HTML). No exemplo a seguir ao clicar no link que está no início da página será direcionado para final da página, pois é lá que se encontra a marcação que possui `id` igual a `inicio`. O símbolo `#` é usado para dizer que `inicio` é um identificador.

```
    <body>
        <h3 id='inicio'>Início da página</h3>
        <a href='#fim'>Link para a marcação que está no fim da página</a>
        <p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p>
        <p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p>
        <p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p>
        <p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p><p>@</p>
        <h3 id='fim'>Fim da página</h3>
        <a href='#inicio'>Link para a marcação que está no início da página</a>
    </body>
```

Um link pode usar uma URL absoluta, isto é, com protocolo (http) e domínio (g1.globo.com) ou uma URL relativa, que parte do caminho atual, por exemplo, se o arquivo atual está em http://localhost/pasta/index.html e colocamos um link para

```
    <a href='teste/teste.txt'>Exemplo 2</a>
```

então o destino será o arquivo http://localhost/pasta/teste/teste.txt.

4 – Imagens no HTML

O elemento `img` é usado para incorporar uma imagem na página. No processo de carregamento da página, primeiramente o navegador carrega do documento HTML e na sequência ele faz a requisição dos recursos incorporados, assim como as imagens. No exemplo a seguir a imagem logo.png é um arquivo a parte, isto é, ele não fica dentro do arquivo HTML.

```
    <img src='logo.png' alt='Google' width="50" height="18" title="Mensagem">
```

O elemento `img` possui apenas a marcação de abertura, e ele requer os atributos `src` (source - URL relativa ou absoluta para o arquivo da imagem) e `alt` (texto a ser exibido se o arquivo da imagem não puder ser localizado / mecanismos de buscas / leitor de telas para deficientes visuais). Os atributos `width` e `height` (em pixels) não são obrigatórios, mas ao especificar eles, a página é renderizada já considerando a largura e altura da imagem a ser carregada, isso faz diferença em redes lentas.

Assim como em qualquer outro elemento HTML, o atributo `title` é usado para mostrar um texto quando o usuário posiciona o cursor sobre o elemento.

5 – Tabelas no HTML

Uma tabela é formada por linhas e colunas, que constituem as células da tabela. Desta forma, o elemento `table` precisa de outros elementos para criar uma tabela.

A tabela é formada por linhas `tr` (table row) e cada linha é formada por `th` (table header) ou `td` (table data), que define respectivamente as células do título e corpo da tabela.

O elemento `th` possui uma formatação em negrito e centralizado horizontalmente. A tabela é uma estrutura rígida, onde todas as linhas precisam ter o mesmo número de colunas e todas as colunas precisam ter o mesmo número de linhas. Para resolver esse problema precisamos usar os atributos `colspan` e `rowspan` para indicar que uma célula ocupa mais de uma coluna e linha respectivamente. No exemplo a seguir o atributo `colspan` foi usado para fazer a célula `<td colspan="3">Total: 4</td>` ocupar o espaço horizontal de 3 células.

Os elementos `thead`, `tbody` e `tfoot` são opcionais. Eles são usados para definir blocos de linhas da tabela, essa estrutura é interessante no momento de aplicarmos formatação, pois os elementos do corpo (`tbody`) podem receber uma formatação diferente dos elementos do cabeçalho (`thead`) e rodapé (`tfoot`).

Como o elemento `table` e os seus sub elementos não possuem a propriedade de borda, então tivemos de usar um estilo CSS para adicionar essa formatação.

Nome	Idade	Escolaridade
Ana	20	Médio
José	28	Superior
Maria	21	Técnico
Pedro	30	Técnico
Total: 4		

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <style>
        table {border:1px solid}
        td, th {border:1px solid}
    </style>
</head>
<body>
    <table>
        <thead>
            <tr>
                <th>Nome</th>
                <th>Idade</th>
                <th>Escolaridade</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Ana</td>
                <td>20</td>
                <td>Médio</td>
            </tr>
            <tr>
                <td>José</td>
```

```
            <td>28</td>
            <td>Superior</td>
        </tr>
        <tr>
            <td>Maria</td>
            <td>21</td>
            <td>Técnico</td>
        </tr>
        <tr>
            <td>Pedro</td>
            <td>30</td>
            <td>Técnico</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="3">Total: 4</td>
        </tr>
    </tfoot>
</table>
</body>
</html>
```

6 – Formulários HTML

O elemento `form` é um envoltório para os elementos que compõem um formulário, tais como, campos de entrada `<input>`, campos de seleção `<select>`, campo para entrada de texto com várias linhas `<textarea>` e botões `<button>`.

O elemento `input` é usado para criar diferentes tipos de marcação, a diferença está no valor do atributo `type`:

- Campo de entrada: o atributo `type` assume o valor padrão, que é `text`. Existem outros valores possíveis, tais como, number, email etc.;
- Radio button: o atributo `type` recebe o valor `radio`. Para formar um grupo o atributo `name` de todos os radio button precisam ter o mesmo valor, no exemplo a seguir, todos eles receberam o valor `doador`;
- Check box: o atributo `type` recebe o valor `checkbox`.

Para mais detalhes sobre o atributo `type` acesse https://www.w3schools.com/tags/tag_input.asp.

Nos campos de entrada o conteúdo do campo se encontra no atributo `value`.

O atributo `placeholder` é usado para exibir um texto no campo de entrada enquanto o atributo `value` estiver vazio.

O elemento `input` com atributo `type` igual a `hidden` tem o objetivo de guardar um valor no HTML, como se fosse uma variável. No exemplo a seguir estamos guardando o valor 10 que pode se acessado a partir usando o atributo `name`.

```
<input type='hidden' name='codigo' value='10'>
```

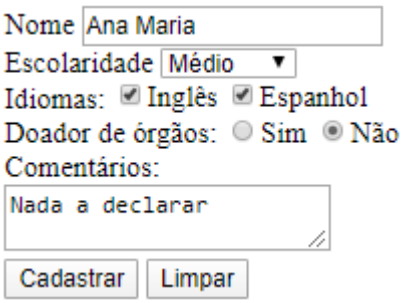
O formulário tem como objetivo a entrada de dados pelo usuário e esses dados podem ser processados na própria página ou envidados para um servidor. Para isso usamos o evento `onSubmit` que está vinculado ao botão com `type` igual a `submit`. Todos os campos do formulário precisam ter o atributo `name` para que eles possam ser submetidos para o servidor.

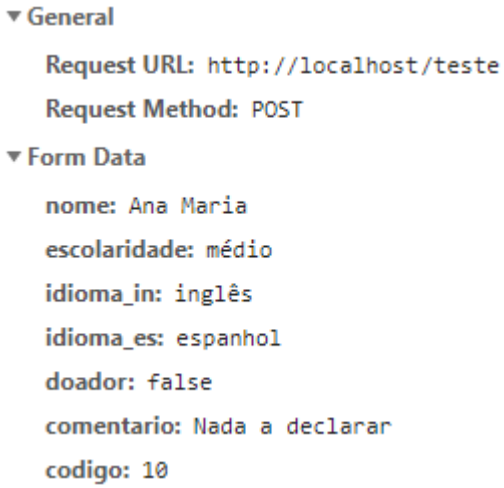
```
<button type="submit">Cadastrar</button>
```

Ao clicar no botão `Cadastrar` o evento `onSubmit` irá fazer o elemento `form` enviar os campos do formulário no formato `name` e `value` para o recurso especificado no atributo `action`, que neste exemplo colocamos o recurso fictício `teste`. Uma requisição HTTP pode ser feita usando os métodos GET, POST, PUT e DELETE. Se não definirmos o atributo `method`, o valor padrão será GET.

```
<form action='/teste' method="post">
```

Como exemplo, o navegador irá fazer a seguinte requisição HTTP ao clicar no botão `Cadastrar` quando o formulário tiver os seguintes dados. Observe que o atributo `name` é usado para identificar os valores a serem enviados.






```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>
    <form action='/teste' method="post">
        <div>
          <label>Nome</label>
          <input type='text' name='nome' placeholder="Nome completo">
        </div>
        <div>
          <label>Escolaridade</label>
          <select name='escolaridade'>
            <option value="médio">Médio</option>
            <option value="técnico">Técnico</option>
            <option value="superior">Superior</option>
          </select>
        </div>
        <div>
            Idiomas:
            <label><input type='checkbox' name='idioma_in' value='inglês'>Inglês</label>
            <label><input type='checkbox' name='idioma_es' value='espanhol'>Espanhol</label>
          </div>
        <div>
```

```
          Doador de órgãos:
          <label><input type='radio' name='doador' value='true' checked='checked'>Sim</label>
          <label><input type='radio' name='doador' value='false'>Não</label>
        </div>
        <div>
            <div>Comentários:</div>
            <textarea name="comentario" placeholder="Digite aqui"></textarea>
        </div>
        <div>
          <button type="submit">Cadastrar</button>
          <button type="reset">Limpar</button>
        </div>
        <input type='hidden' name='codigo' value='10'>
      </form
</body>
</html>
```

vii. Linguagem de formatação CSS (Cascading Style Sheets)

Na página web usamos as marcações HTML para criar a estrutura do documento, isto é, constitui boa prática não usar marcações de formatação para formatar o documento HTML. Para definir a formatação dos elementos HTML usamos a linguagem de estilos CSS.

A linguagem CSS é formada por propriedades que podem ser aplicadas aos elementos HTML. A definição de uma propriedade CSS é formada pelo par nome e valor:

```
name:value
```

Existem muitas propriedades, mas aqui apresentaremos apenas alguns exemplos para verificarmos a sua regra de construção:

- Propriedades que possuem somente 1 valor. No exemplo a seguir a propriedade `color` recebe somente o valor `yellow`.

  ```
  color: yellow
  ```

  O valor da propriedade color pode ser definido de diferentes formas. Nos exemplos a seguir todas as cores são o mesmo amarelo:

  ```
  color: yellow;
  color: #ffff00;
  color: rgb(255, 255, 0);
  ```

  Na notação hexadecimal usa-se #RedRedGreenGreenBlueBlue e na função rgb os parâmetros possuem o seguinte significado rgb(red, green, blue), em ambos os casos os valores variam entre [0,255].

  Para mais detalhes acesse https://www.w3schools.com/cssref/css_colors_legal.asp

- Propriedade com nome composto: usa-se hífen para compor o nome da propriedade que possui mais de um nome, pois assim como nas linguagens de programação, um nome de recurso não pode ter espaços:

  ```
  background-color: yellow;
  font-size: 14px;
  text-align: center;
  ```

- Propriedade com valor composto: no exemplo a seguir a borda irá receber 1 pixel de largura, será do tipo contínua e vermelha, isto é, podemos definir as três propriedades da borda usando espaços:

```
border: 1px solid red
```

As propriedades de `border` podem ser definidas individualmente:

```
border-width: 2px;
border-style: dotted;
border-color: blue;
```

A propriedade `border` define a borda nos quatro lados (top, right, bottom e left), porém podemos definir as bordas individualmente:

```
border-top: 1px solid red;
border-right: 2px solid green;
border-bottom: 1px dotted blue;
border-left: 4px solid yellow;
```

Da mesma forma podemos especificar apenas a cor, estilo e largura de um lado da borda:

```
border-top-color: red;
```

Para mais detalhes https://www.w3schools.com/css/css_border.asp

- Propriedades com sequência de valores: a propriedade `font-family` pode receber vários nomes de fontes, mas ela usará apenas uma fonte. No exemplo a seguir, o navegador irá verificar se existe a fonte `Arial`, se ela não existir o navegador usará a fonte `Helvetica`, e se não tiver ele usará `sans-serif`, e se não tiver ele usará a fonte padrão do navegador. Os nomes são separados por vírgula.

```
font-family: Arial, Helvetica, sans-serif;
```

Adicionar CSS no documento HTML

O CSS pode ser aplicado no documento HTML de três formas:

1. Inline: o estilo CSS é aplicado somente na própria marcação. O atributo `style` é usado para receber as propriedades CSS do elemento. No exemplo a seguir as formatações são aplicadas apenas ao próprio elemento.

   O CSS inline deve ser usado muito pouco no documento, pois não é recomendado misturar a estrutura do documento com a sua formatação.

   Observe que as propriedades são separadas por ponto e vírgula.

```
<p style='color:red;background-color:yellow'>bom dia</p>
<p style='font-size:20px'>boa tarde</p>
```

bom dia

boa tarde

2. Incorporado ou interno: as propriedades CSS são declaradas no elemento `style`, que está no cabeçalho do documento `<head>`. Para aplicar a formatação aos elementos específicos do documento precisamos vincular as formatações aos seletores de elementos HTML – posteriormente será explicado sobre seletores CSS. No exemplo a seguir todos os elementos `p` do documento irão receber a mesma formatação.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <style>
      p {
        color:red;
        background-color:yellow
      }
      div {
        font-size:20px;
        color:green
      }
    </style>
  </head>
  <body>
    <p>bom dia</p>
    <p>boa tarde</p>
    <div>boa noite</div>
  </body>
</html>
```

bom dia

boa tarde

boa noite

3. Vinculado ou externo: as propriedades CSS estão num arquivo `.css` separado e esse arquivo é vinculado ao documento HTML através do elemento `link`. Esse elemento precisa estar no cabeçalho do documento `<head>` e o atributo `href` recebe a URL relativa ou absoluta do arquivo a ser vinculado. O atributo `rel` diz ao navegador que o link é para um arquivo de folhas de estilo (CSS).

   A vinculação de arquivo .css é a forma mais habitual e útil para estruturar o CSS, pois a mesma formatação pode ser aplicada nos vários documentos HTML de um portal.

Arquivo: ex1.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="ex1.css">
  </head>
  <body>
    <p>bom dia</p>
    <p>boa tarde</p>
    <div>boa noite</div>
  </body>
</html>
```

Arquivo: ex1.css

```
p {
    color:blue;
    background-color:lightgray
}
div {
    border-left:4px solid lime;
    font-weight: bold
}
```

Resultado

bom dia

boa tarde

boa noite

Seletores CSS

Os seletores CSS são usados para atingir os elementos do documento HTML. Existem vários tipos de seletores, aqui iremos apresentar apenas os mais usados, veja no exemplo a seguir.

Arquivo: ex1.html	Arquivo: ex1.css	Resultado

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet"  href="ex1.css">
  </head>
  <body>
    <p>um</p>
    <p id='principal'>dois</p>
    <p class='destaque extra'>três</p>
    <div>quatro</div>
    <div class='destaque'>cinco</div>
    <div class='extra'>seis</div>
  </body>
</html>
```

```css
#principal {
    color:magenta;
    font-style: italic;
}
.destaque {
    color:green;
    font-weight: bold;
}
.extra {
    text-decoration: underline;
}
p {
    color:red;
}
div {
    color:blue;
}
p, div {
    font-family: Arial;
    font-size: 15px;
}
```

um

dois

três

quatro
cinco
seis

- Seletor por tagname: seleciona todos os elementos do documento que possuem o tagname especificado. Por exemplo, todos os elementos `p` irão receber o estilo de cor da fonte vermelha:

```
p { color:red }
```

- Seletor por classe (`. ponto`): seleciona todos os elementos do documento que possuem o valor do atributo `class`. Por exemplo, todos os elementos que possuem `class='destaque'` serão selecionados pelo seletor a seguir.

```
.destaque {color:green; font-weight: bold}
```

Podem existir vários elementos no documento com a mesma classe, neste exemplo foram apenas dois:

```
<p class='destaque extra'>três</p>
<div class='destaque'>cinco</div>
```

Um elemento pode ter mais de uma classe, mas elas precisam estar separadas pelo caractere de espaço:

```
<p class='destaque extra'>três</p>
```

- Seletor por identificador `#`: seleciona o elemento do documento que possui o valor do atributo `id`. Por exemplo, será selecionado o elemento que possui `id='principal'`.

```
#principal {color:magenta; font-style: italic}
```

Deve-se ter apenas um elemento do documento com o mesmo valor do atributo `id` (identificador precisa ser único).

Podemos aplicar um estilo para mais de um seletor, mas neste caso os seletores devem estar separados por vírgula. No exemplo seguir os elementos `p` e `div` receberão o estilo definido entre as chaves:

```
p, div { font-family: Arial; font-size: 15px; }
```

Layouts - Elementos block box e inline box

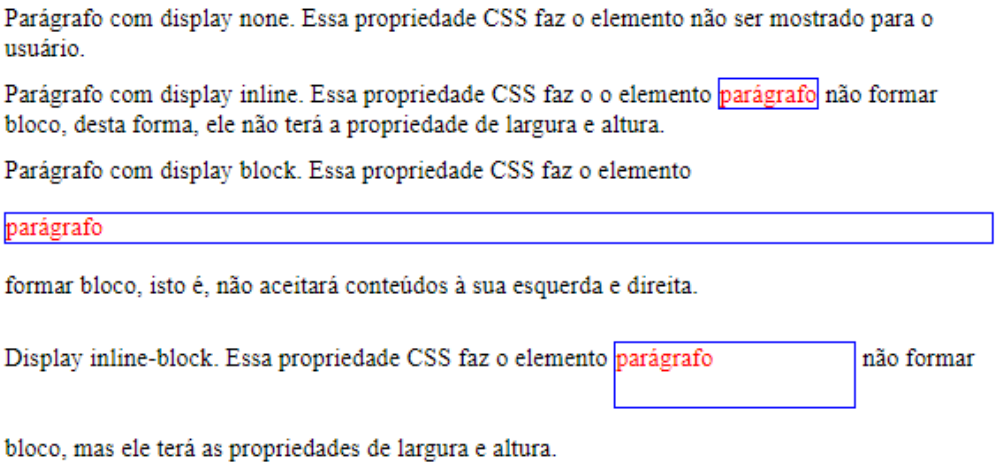
No CSS existem dois tipos de blocos, a diferença entre eles está apenas no comportamento deles no fluxo da página:

- Block box: blocos que formam boxes (caixas), isto é, ocupam 100% da área horizontal disponível para ele, e, desta forma, não aceitam outros elementos à sua esquerda e direita. Exemplos de elementos que criam blocos: `<p>`, `<div>` e `<h1>`. Sobre as propriedades:
  - `width` e `height` são respeitadas num elemento que forma bloco;
  - `padding`, `margin` e `border` são computadas fazendo com que a largura/altura do componente ultrapasse 100%.
- Inline box: ocupam apenas a sua própria área e, desta forma, aceitam outros elementos à sua esquerda e direita. Exemplos de elementos inline: `<a>`, `<span>`, `<em>` e `<strong>`.

  Sobre as propriedades:
  - `width` e `height` não são respeitadas num elemento inline;
  - `padding`, `margin` e `border` são aplicadas, mas não fazem com que outros elementos inline sejam deslocados para a linha seguinte.

O tipo block ou inline pode ser definido pela propriedade CSS `display` do elemento. No exemplo a seguir o elemento `p`, que possui por padrão as propriedades de block box, mudou o comportamento de acordo com a propriedade `display`.

Resultado



Arquivo: ex1.html	Arquivo: ex1.css

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="ex1.css">
</head>
<body>
  <div>
    Parágrafo com display none. Essa propriedade CSS
   faz o elemento <p class="ex1">parágrafo</p> não ser
   mostrado para o usuário.
  </div>
  <div>
    Parágrafo com display inline. Essa propriedade CSS
    faz o elemento <p class="ex2">parágrafo</p> não formar
    bloco, desta forma, ele não terá a propriedade de
    largura e altura.
  </div>
  <div>
    Parágrafo com display block. Essa propriedade CSS faz
    o elemento <p class="ex3">parágrafo</p> formar
    bloco, isto é, não aceitará conteúdos à sua esquerda e direita.
  </div>
  <div>
    Display inline-block. Essa propriedade CSS faz o elemento
   <p class="ex4">parágrafo</p> não formar bloco, mas
   ele terá as propriedades de largura e altura.
  </div>
</body>
</html>
```

```css
div {margin-bottom: 10px}
p {color: red;}
.ex1 {
    display: none;
    border:1px solid blue
}
.ex2 {
    display: inline;
    border:1px solid blue;
    width:150px;
    height:40px
}
.ex3 {
    display: block;
    border:1px solid blue
}
.ex4 {
    display: inline-block;
    border:1px solid blue;
    width:150px; height:40px
}
```

Box Model

O box model completo se aplica aos elementos block box, os elementos inline box usam apenas parte do comportamento definido no box model. O modelo define como as diferentes partes de um box (caixa) - margem, borda, preenchimento e conteúdo - trabalham juntas para criar um box na página. A Figura 1 mostra as partes do box:

- Content box: área onde o conteúdo é exibido, que pode ser dimensionado usando as propriedades CSS de width e height;
- Padding box (caixa de preenchimento): o preenchimento fica em torno do conteúdo como espaço em branco (como se fosse uma margem interna). O seu tamanho pode ser controlado usando a propriedade CSS padding;
- Border box: a caixa de borda envolve o conteúdo e qualquer preenchimento. Seu tamanho e estilo podem ser controlados usando a propriedade CSS border;
- Margin box: a margem é a camada mais externa, envolvendo a borda como espaço em branco entre o próprio elemento e os demais. Seu tamanho pode ser controlado usando a propriedade CSS margin.

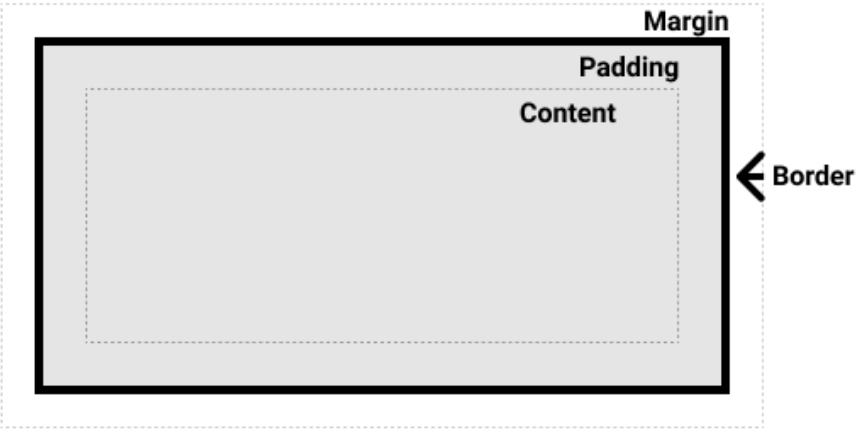


Figura 1 – Conteúdo, padding, border e margin de um elemento HTML.

Fonte: https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model

O exemplo a seguir mostra que as propriedades margin, border e padding são computadas nas dimensões final do elemento na página.

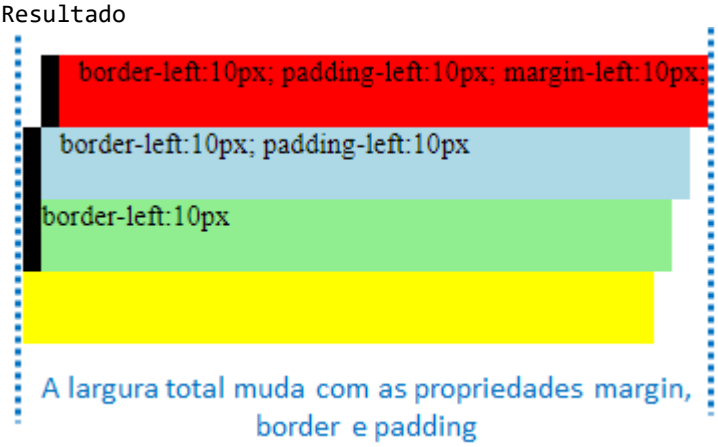
Arquivo: ex1.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="ex1.css">
</head>
<body>
  <div class='vermelho'>border-left:10px;
                  padding-left:10px; margin-left:10px;</div>
  <div class='azul'>border-left:10px; padding-left:10px</div>
  <div class='verde'>border-left:10px</div>
  <div class='amarelo'></div>
</body>
</html>
```

Arquivo: ex1.css

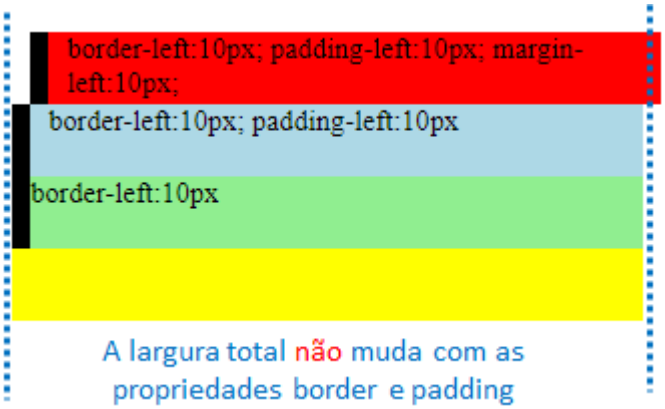
```
.vermelho {
    border-left: 10px solid black;
    padding-left:10px;
    margin-left:10px;
    background-color:red
}

.azul {
    border-left: 10px solid black;
    padding-left:10px;
    background-color:lightblue
}

.verde {
    border-left: 10px solid black;
    background-color:lightgreen
}

.amarelo {
    background-color:yellow
}

.vermelho, .azul, .verde, .amarelo {
    width:350px;
    height:40px;
}
```

Resultado



Os navegadores usam o box model padrão, mas se usarmos a propriedade `box-sizing: border-box` o comprimento da borda e padding são subtraídos do conteúdo. No exemplo a seguir, adicionamos apenas a propriedade `box-sizing: border-box` no código do exemplo anterior.

```
.vermelho, .azul, .verde, .amarelo {
    width:350px;
    height:40px;
    box-sizing: border-box;
}
```



Observação: se quisermos usar o box model alternativo em toda a página, basta setar a propriedade `box-sizing: border-box` no elemento `<html>`, e fazer os elementos filhos herdarem essa propriedade. Veja a seguir as modificações necessárias no código. O seletor `*` é usado para selecionar todos os elementos do documento.

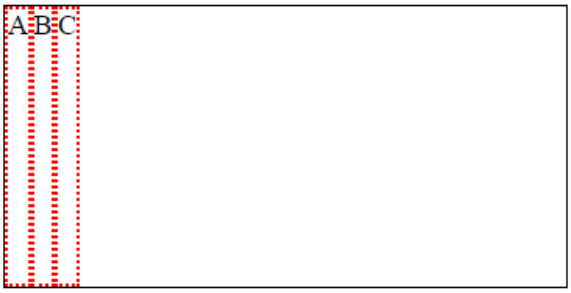
```
.vermelho, .azul, .verde, .amarelo {
    width:350px;
    height:40px;
}

html {
    box-sizing: border-box;
}

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

viii. Flexbox layout

O Flexbox ajuda a construir layouts responsivos. Para criar um layout responsivo precisamos definir um elemento pai com a propriedade `display:flex`, desta forma, os elementos filhos serão flexíveis.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .um {
            display: flex;
            border: 1px solid black;
            height: 200px;
            width: 400px;
        }
        .um>div {
            border: 2px dotted red;
```

Resultado:



```
            font-size: 20px;
        }
    </style>
</head>
<body>
    <div class="um">
        <div>A</div>
        <div>B</div>
        <div>C</div>
    </div>
</body>
</html>
```

Quando os elementos são definidos como flex boxes, eles são dispostos ao longo de dois eixos (Figura 2):

- O main axis é o eixo que corre na direção em que os flex items estão dispostos (por exemplo, as linhas da página, ou colunas abaixo da página). O início e o fim do eixo são chamados de main start e main end;
- O cross axis é o eixo perpendicular que corre na direção em que os flex items são dispostos. O início e o fim deste eixo são chamados de cross start e cross end;
- O elemento pai que possui display:flex é chamado de flex container;
- Os itens iniciados como flexible boxes dentro do flex container são chamados flex items.

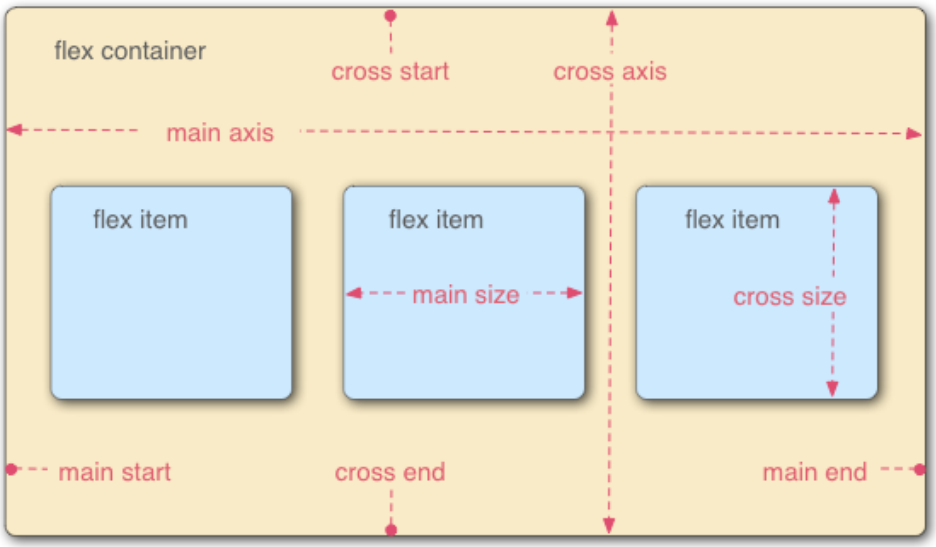


Figura 2 – Disposição dos elementos flexbox nos eixos principal e transversal.

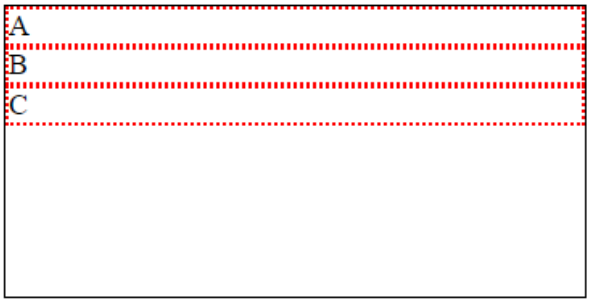
Fonte: https://developer.mozilla.org/pt-BR/docs/Learn/CSS/CSS_layout/Flexbox

Flexbox possui a propriedade `flex-direction` que especifica a direção do eixo principal - isto é, a direção que os filhos do flexbox estarão arranjados - que por padrão é `row` (linha). No exemplo a seguir adicionamos a `flex-direction: column` no componente pai para mudar a ordem de `row` para `column`.

Resultado:

```
<!DOCTYPE html>
<html>
<head>
    <style>
```

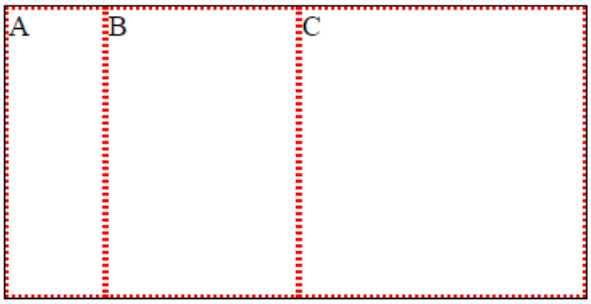
```
        .um {
            display: flex;
            flex-direction: column;
            border: 1px solid black;
            height: 200px;
            width: 400px;
        }
        .um>div {
            border: 2px dotted red;
            font-size: 20px;
        }
    </style>
</head>
<body>
    <div class="um">
        <div>A</div>
        <div>B</div>
        <div>C</div>
    </div>
</body>
</html>
```



A propriedade flex pode ser usada nos elementos filhos para controlar a proporção de espaço que os flex items podem ocupar. Na prática podemos usar quaisquer valores na propriedade flex, no exemplo a seguir os elementos a A, B e C não possuem a mesma largura porque colocamos proporções diferentes: A irá ocupar 1/(1+2+3), B irá ocupar 2/(1+2+3) e C irá ocupar 3/(1+2+3). Cada elemento irá ocupar a mesma largura se definirmos flex:1 em todos eles elementos filhos.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .um {
            display: flex;
            border: 1px solid black;
            height: 200px;
            width: 400px;
        }
        .um>div {
            border: 2px dotted red;
            font-size: 20px;
        }
    </style>
</head>
<body>
    <div class="um">
        <div style="flex:1">A</div>
        <div style="flex:2">B</div>
        <div style="flex:3">C</div>
    </div>
```

Resultado:

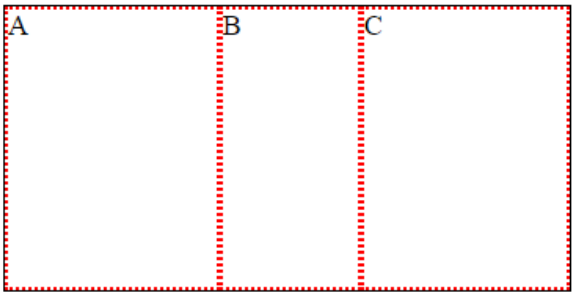


```
</body>
</html>
```

A propriedade flex aceita uma largura mínima para o elemento filho. No exemplo a seguir o elemento A irá ocupar o mínimo de 100px. O restante da largura será dividido proporcionalmente entre os elementos B e C.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .um {
            display: flex;
            border: 1px solid black;
            height: 200px;
            width: 400px;
        }
        .um>div {
            border: 2px dotted red;
            font-size: 20px;
        }
    </style>
</head>
<body>
    <div class="um">
        <div style="flex:1 100px">A</div>
        <div style="flex:2">B</div>
        <div style="flex:3">C</div>
    </div>
</body>
</html>
```

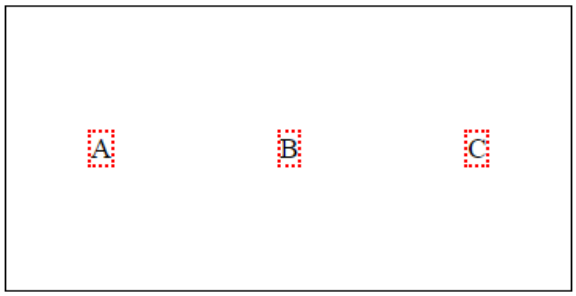
Resultado:



As propriedades align-itens e justify-content são usadas, respectivamente, para alinhar no eixo transversal e principal (neste exemplo, o eixo principal é row).

```
<html>
<head>
    <style>
        .um {
            display: flex;
            align-items: center;
            justify-content: space-around;
            border: 1px solid black;
            height: 200px;
            width: 400px;
        }
        .um>div {
            border: 2px dotted red;
            font-size: 20px;
        }
    </style>
```

Resultado:



```
</head>
<body>
    <div class="um">
        <div>A</div>
        <div>B</div>
        <div>C</div>
    </div>
</body>
</html>
```

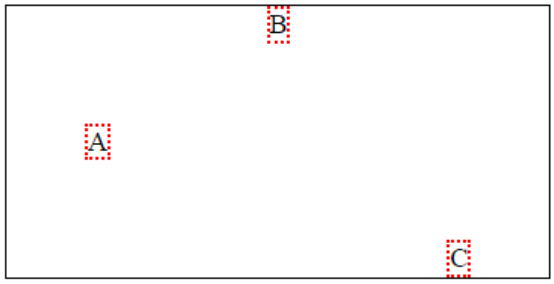
A propriedade `align-itens` controla onde os elementos flex ficam no eixo transversal:

- `stretch`: estica todos os elementos flex para preencher o elemento pai na direção do eixo transversal. Se o elemento pai não tem largura fixa na direção do eixo transversal, então todos os elementos flex esticarão até o mais comprido dos elementos flex;
- `center`: faz com que os elementos mantenham suas dimensões intrínsecas, mas que sejam centralizados ao longo do eixo transversal;
- `flex-start` e `flex-end`: alinha todos os elementos no início ou fim do eixo transversal, respectivamente.

A propriedade `align-self` sobrescreve a propriedade `align-items` para elementos individuais. No exemplo a seguir os elementos filhos possuem alinhamentos distintos daquele especificado no elemento pai.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .um {
            display: flex;
            align-items: center;
            justify-content: space-around;
            border: 1px solid black;
            height: 200px;
            width: 400px;
        }
        .um>div {
            border: 2px dotted red;
            font-size: 20px;
        }
    </style>
</head>
<body>
    <div class="um">
        <div>A</div>
        <div style="align-self: flex-start">B</div>
        <div style="align-self: flex-end">C</div>
    </div>
</body>
</html>
```

Resultado:



A propriedade `justify-content` controla onde os elementos flex ficam no eixo principal.

- `flex-start` e `flex-end`: todos os elementos estejam no início ou final do eixo principal, respectivamente;
- `center`: todos os elementos flex fiquem no centro do eixo principal;
- `space-around`: distribui todos os elementos igualmente pelo eixo principal, com um pouquinho de espaço no início e final;
- `space-between`: similar ao space-around, exceto que ele não deixa nenhum espaço no início e final.