CEUB

EDUCAÇÃO SUPERIOR

Desenvolvimento de Interfaces - DI

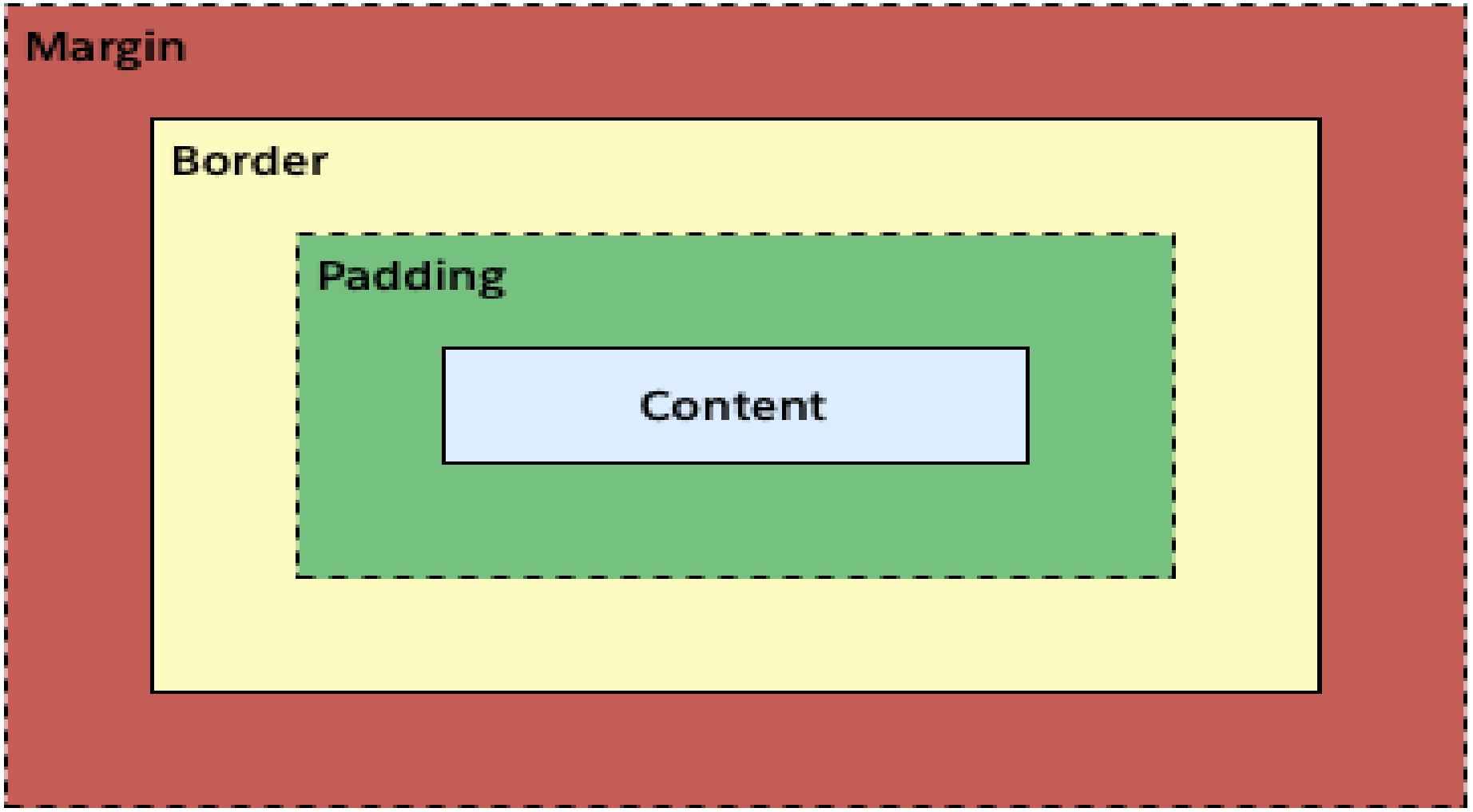
ceub.br

CEUB
EDUCAÇÃO SUPERIOR

Aula 05 – CSS Bordas e Alinhamento

Bordas e Alinhamentos CSS

CSS Box Model

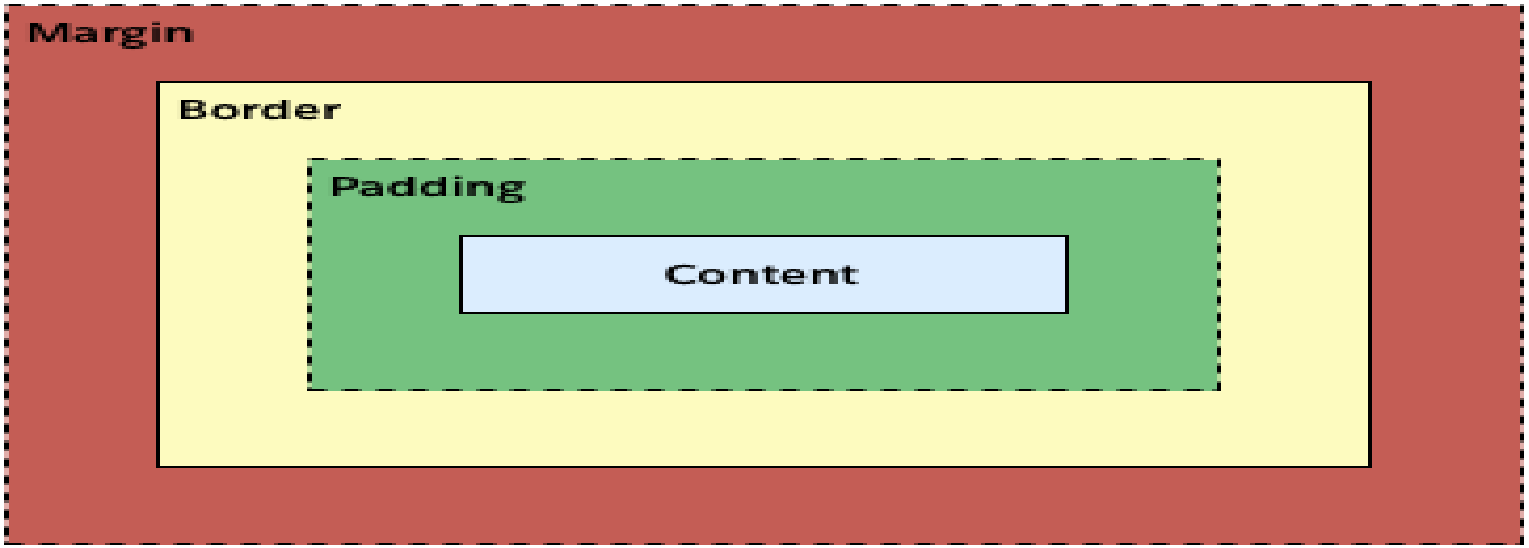


CSS Box Model

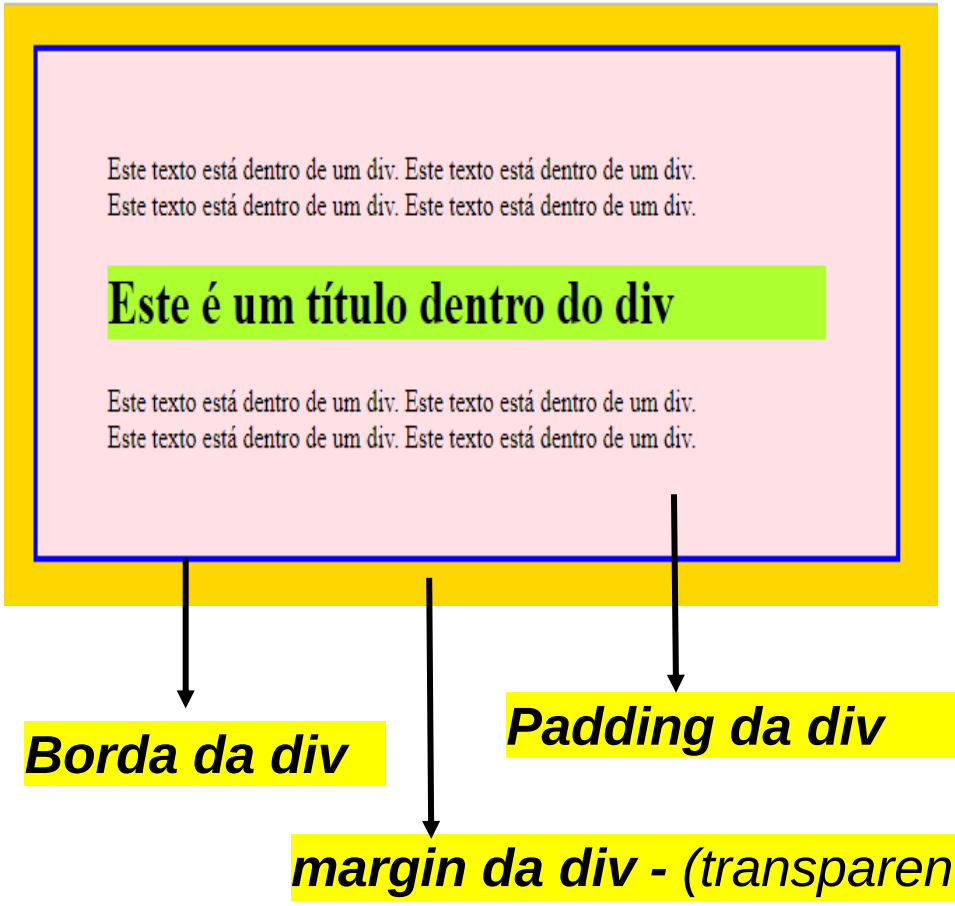
Todos os elementos HTML podem ser considerados como caixas. No CSS, o termo "modelo de caixa" é usado quando se fala em design e layout.

O modelo de caixa CSS é essencialmente uma caixa que envolve todos os elementos HTML. Consiste em: margens, bordas, preenchimento e o conteúdo real.

A imagem abaixo ilustra o modelo da caixa:



CSS - Margin x Padding x Border




```
<style>
    body {
        background-color: gold;
    }

    h1 {
        background-color: greenyellow;
    }

    div {
        background-color: #ffe0e5;
        margin: 20px;
        padding: 50px;
        border: 3px solid blue;
    }
</style>
```

Exemplo – Criar uma página com o css, que reproduza a tela acima.
Ver página ex. 4

Propriedade *Margin*

- A margem é a área ao redor (fora da borda) do elemento HTML;
- A margem não tem cor de fundo (é transparente);
- É possível especificar as margens superior, inferior, esquerda e direita individualmente (ou todas de uma vez);
- Exemplo de definição individual das margens:

```
p {
   margin-top: 150px;
   margin-bottom: 150px;
   margin-right: 200px;
   margin-left: 100px;
}
```

Propriedade *Margin*

É possível definir todas as margens (sup, dir, inf, esq) de uma só vez utilizando a propriedade abreviada *margin*:

Exemplos:

```
margin: 25px 50px 75px 100px;
```
(superior, direita, inferior e esquerda, respectivamente)

```
margin: 25px 50px 75px;
```
(superior; esquerda-direita; inferior, respectivamente)

```
margin: 25px 50px;
```
(superior e inferior; esquerda e direita; respectivamente)

```
margin: 25px;
```
(todas as margens de 25px)

Exercício: testar os ajustes de margem utilizando o exemplo anterior

Propriedade *Padding*

- É a área em volta do conteúdo do elemento (dentro da borda);
- O padding é afetado pela cor de fundo (background-color) do elemento;
- O padding superior, inferior, esquerdo e direito podem ser especificados individualmente ou todos de uma vez (como na definição das margens);

Exemplos:

```
p {
    margin-top:150px;
    margin-bottom:150px;
    margin-right:200px;
    margin-left: 100px;
}
div {
   padding: 70px 100px 70px 50px;
}
```

Propriedade *Padding*

- É a área em volta do conteúdo do elemento (dentro da borda);
- O padding é afetado pela cor de fundo (background-color) do elemento;
- O padding superior, inferior, esquerdo e direito podem ser especificados individualmente ou todos de uma vez (como na definição das margens);

Exemplos:

```
p {
    margin-top:150px;
    margin-bottom:150px;
    margin-right:200px;
    margin-left: 100px;
}
div {
   padding: 70px 100px 70px 50px;
}
```

Bordas

Uma forma rápida e prática de definir as bordas inferior, superior, esquerda e direita de um elemento HTML é utilizar a propriedade *border*

border: ***espessura estilo cor***

Exemplo:

```
div {
    border: 2px solid green;
}
```

Para definir apenas uma das bordas ou bordas com estilos diferentes, pode-se utilizar as propriedades:

- ***border-left***
- ***border-right***
- ***border-top***
- ***border-bottom***

Bordas

É possível definir os estilos da borda (cor, largura, etc) de maneira separada ou específica.

- **border-style:** define o estilo da borda (none, solid, dotted, dashed, double)
- **border-width:** define a espessura da borda;
- **border-color:** define a cor da borda;
- **border-radius:** define a borda com cantos arredondados.

EX

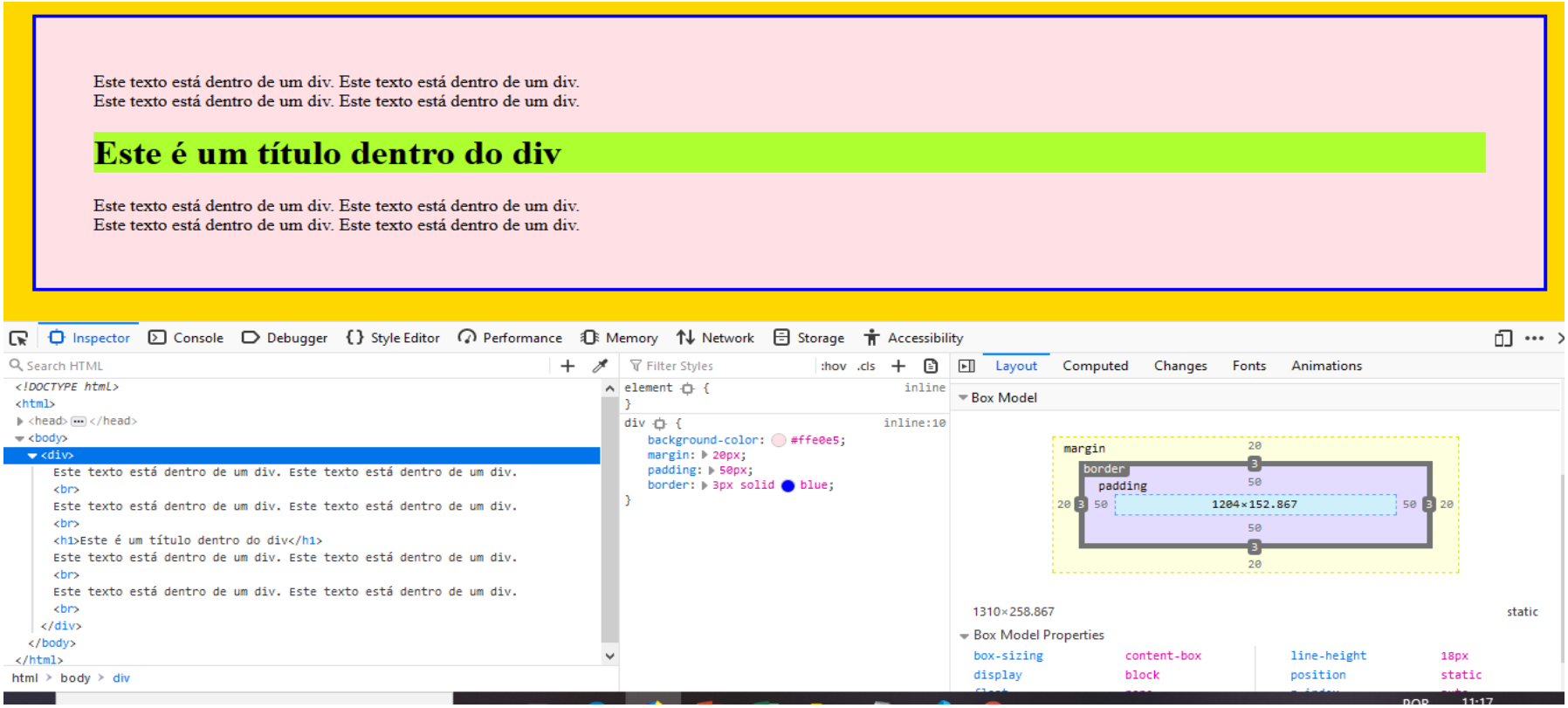
```
h1 {
    border-style: solid; border-color: rgb(245, 16, 54); border-width: 4px;
 }
```

Ex: borda específica

```
p {
    border-top-style: dotted;
    border-right-style: solid;
    border-bottom-style: dotted;
    border-left-style: solid;
}
```

Inspecionando os Elementos da Página

Em vários navegadores é possível verificar (e até mesmo modificar), em tempo real, os valores atribuídos à margem, borda e padding de alguns elementos HTML; No Firefox , clique sobre o elemento com o botão direito e escolha Inspecionar. Em seguida, explore os recursos.



Width e Height

Utilizadas para definir, respectivamente, a largura e a altura de um elemento;

Podem ser definidas para o valor auto (automático, calculado pelo navegador), valores em pixels (px) e porcentagem.

Exemplo:

```
img {
    width: 200px;
    height: 100px;
}
div {
    /*ocupará 50% da largura disponivel */
    width: 50%;
    height: 30px;
}
body {
    width: 50%;
    margin: 0 auto;
}
```

Width – Largura da página

Uma forma de definir a largura do conteúdo principal da página é utilizando a propriedade width do elemento body;

Para que o corpo da página apareça centralizado é necessário definir também as margens laterais como auto;
Exemplo

```
<style>
    body {
        width: 80%;
        margin: 0 auto;
    }
</style>
```

Width – Largura da página

Exemplo

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <style>

    body {
        background-color: #EEE;
    width: 60%;
    margin: 10px auto;
    }

    </style>
</head>
```

```
<body>

<h1>Página com conteúdo centralizado</
h1>
<p>
  Lorem,.....
</p>

<h1>Página com conteúdo centralizado</
h1>
<p>
   lorem
</p>

</body>
</html>
```

Elementos Block vs Inline

Um elemento com apresentação em bloco (block) toma toda a largura disponível, com quebra de linha antes e depois. Exemplos:

- **<h1>**
- **<p>**
- **<li>**
- ** **

Um elemento com apresentação em linha (inline) toma apenas o espaço necessário para a sua exibição (e sem quebra de linha); Exemplos:

- ****
- **<a>**
- **<img>**

Elementos Block vs Inline

Um elemento de bloco (como o) pode ser centralizado horizontalmente definindo-se as margens laterais com o valor auto (de automático)

Exemplo:

```
margin: 20px auto;
```

Entretanto, para evitar que o elemento ocupe todo o espaço disponível, utilize a definição anterior em conjunto com a propriedade width;

Para apenas centralizar o texto dentro do elemento, utilize text- align:center;

Elementos Block vs Inline

Exemplo

```
<style>
  body {
    background-color: chartreuse;
    font-family: sans-serif;
    color:blueviolet;
  }

  div {
    width: 60%;
    margin: 20px auto;
    padding: 20px 40px;
    background-color: white;
    border: 0.5px solid lightgray;
    border-radius: 10px;
  }
</style>
```

```
<body>

 <div>

   <h1>Conteúdo centralizado</h1>
    <p> Lorem ipsum dolor
    </p>
   h1>Conteúdo centralizado</h1>
   /h1>
    <p>
      Lorem....
    </p>

  </div>

</body>
```

Display

A propriedade CSS display especifica o tipo de caixa de renderização usada por um elemento.

- **No HTML, os valores padrões da propriedade display são feitas a partir do comportamento descrito nas especificações HTML ou da folha de estilo padrão do navegador/usuário.**
- **O valor padrão em XML é inline.**
- **Além dos muitos tipos diferentes de exibição de caixa, o valor none permite desativar a exibição de um elemento; quando você usa none, todos os elementos descendentes também tem a sua exibição desativada.**
- **O documento é renderizado como se o elemento não existisse na árvore do documento.**

Display

O tipo de apresentação block ou inline pode ser alterado com a propriedade display;

O exemplo abaixo altera o modo de apresentação dos itens de lista para inline. Após esta alteração, os itens de uma lista seriam exibidos na mesma linha;

```
li {
   display: inline;
}
```

A propriedade display também é comumente utilizada para ocultar elementos na página. Neste caso, deve-se utilizar:
'display: none';

Display

OBS 1: Um outro valor comum para a propriedade *display* é inline-block.

Com inline-block é possível definir a largura e a altura do elemento (utilizando width e height).

Além disso, com inline-block, as margens e *paddings* superiores e inferiores são respeitadas (com *inline*, não).

OBS 2: Outra forma de ocultar um elemento é utilizando a propriedade *visibility:hidden*.

Desta forma, o navegador oculta o elemento mas mantém o espaço ocupado pelo mesmo vazio.

Com display:none o espaço do elemento ocultado pode ser ocupado por outros elementos.

Float

Propriedade float é muito utilizada para posicionar elementos lado a lado. Por exemplo, é possível organizar parágrafos de texto ao redor de imagens, permitindo que a imagem "flutue" sobre o texto;

Exemplo:

```
<!DOCTYPE html>
<html>
<head>
     <meta charset="UTF-8">
     <style>
          body {
               width: 80%;
               margin: 0 auto;
               background-color: #EEE;
          }
          img {
               border-radius: 5px;
               margin-right: 10px;
               float: left;
          }
     </style>
</head>
<body>
     <img src="<a href="https://www.uniceub.br/">UniCEUB
</a>" width="100px" height="100px">
     <p>UniCeub, UNICEUB, UNICEUB, UNICEUB, UNICEUB, UNICEUB. </p>
     <p>UNICEUB, UNICEUB, UNICEUB, UNICEUB, UNICEUB, UNICEUB. </p>
     <p>UNICEUB, UNICEUB, UNICEUB, UNICEUB,UNICEUB, UNICEUB.  </p>
     <p>UNICEUB, UNICEUB, UNICEUB, UNICEUB,UNICEUB, UNICEUB.  </p>
</body>
</html>
```

Float

Exemplo2:

```html
<!DOCTYPE html>
<html lang="en-us">
<head>
<style>
.city {
    float: left;
    margin: 5px;
    padding: 15px;
    width: 300px;
    height: 300px;
    border: 1px solid black;
}
</style>
</head>
<body>

<h1>Div float left</h1>

<div class="city">
  <h2>Londres</h2>
  <p>Londres é a capital da Inglaterra.</p>
  <p>É a cidade mais populosa do Reino Unido, com uma área metropolitana de mais de 13 milhões de habitantes.</p>
</div>

<div class="city">
  <h2>Paris</h2>
  <p>Paris é a capital da França.</p>
  <p>Paris é um dos maiores centros populacionais da Europa, com mais de 12 milhões de habitantes.</p>
</div>

<div class="city">
  <h2>Tóquio</h2>
  <p>Tóquio é a capital do Japão.</p>
  <p>Tóquio á a área metropolitana mais populosa do mundo</p>
</div>

<div class="city">
  <h2>New York</h2>
  <p>A cidade de Nova York é a cidade mais populosa dos Estados Unidos.</p>
  <p>Nova York é um importante centro de diplomacia internacional e tem sido descrita como a capital cultural e financeira do mundo.</p>
</div>

</body>
</html>
```

Overflow

Propriedade overflow define como o conteúdo de um elemento deve ser tratado quando ele extrapolar a borda do elemento;

Valores possíveis:

- **visible (default)**
- **hidden**
- **scroll**
- **auto**

Abra o arquivo exemploOverflow.css no navegador e verifique o conteúdo do div.

Em seguida, altere o valor da propriedade overflow no div com os diferentes valores indicados acima e observe o resultado.

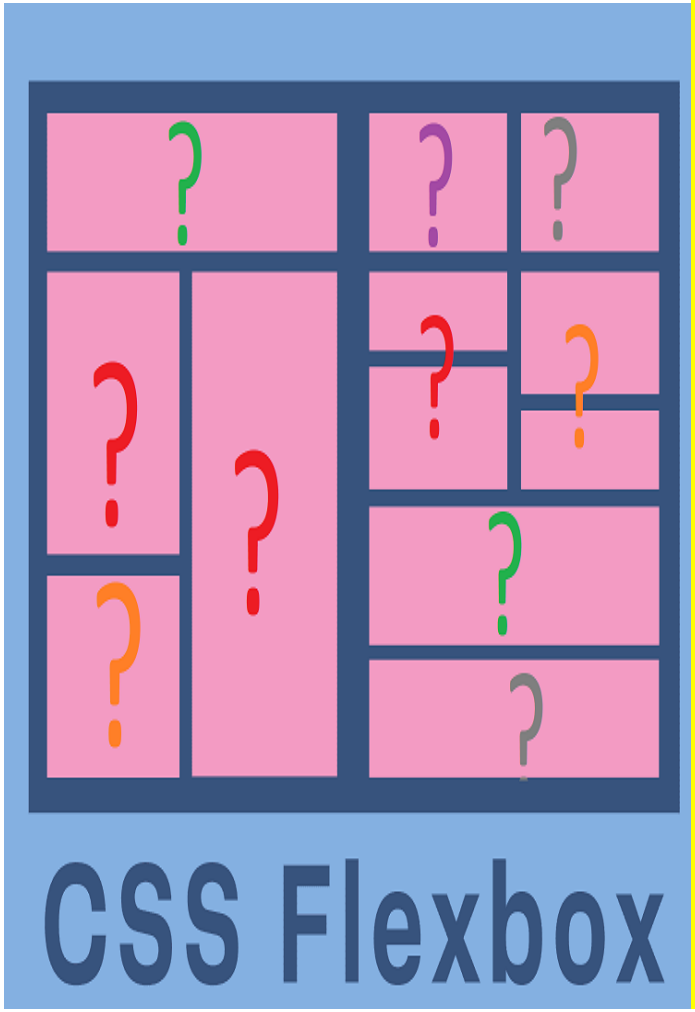
Flexbox

Flexbox

O que é Flexbox?

- **Flexbox, ou "Flexible Box Layout," é um modelo de layout em CSS projetado para facilitar a criação de layouts complexos e flexíveis em uma página da web.**
- **Ele especialmente útil para criar layouts responsivos e alinhar elementos em uma única dimensão (linha ou coluna).**

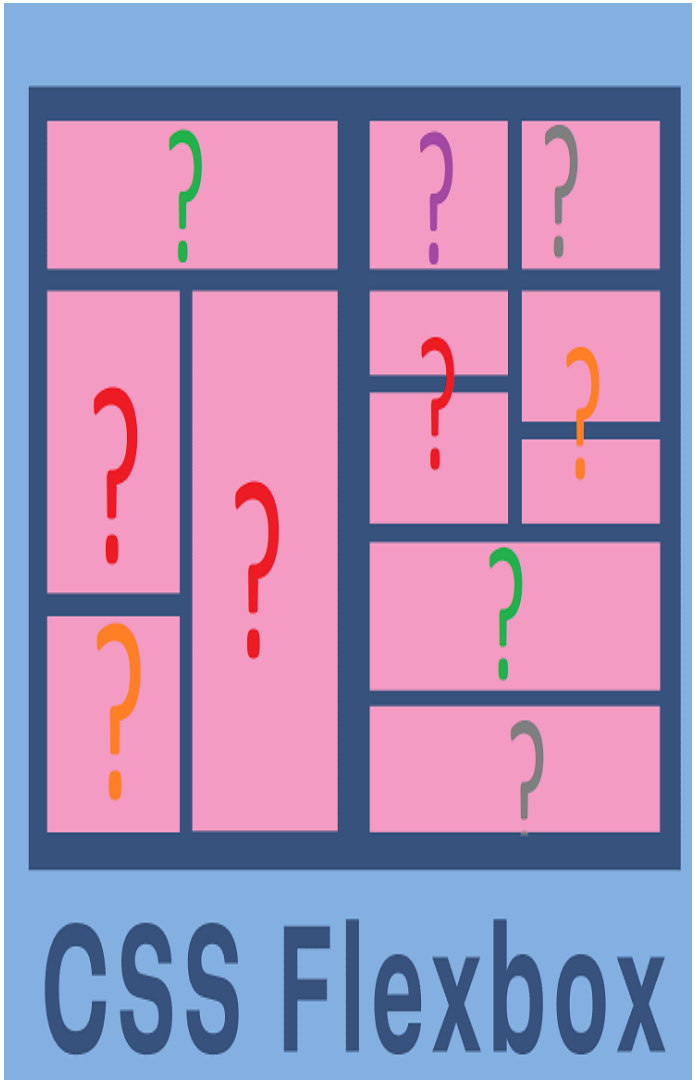
Flexbox



Por que usar o Flebox?

- **Layouts Flexíveis e Responsivos**: **Flexbox permite criar layouts que se ajustam automaticamente a diferentes tamanhos de tela e dispositivos, tornando seu site ou aplicativo mais adaptável.**
- **Alinhamento Simplificado**: **Ele facilita o alinhamento vertical e horizontal de elementos, economizando tempo e código em comparação com métodos mais antigos de alinhamento em CSS.**

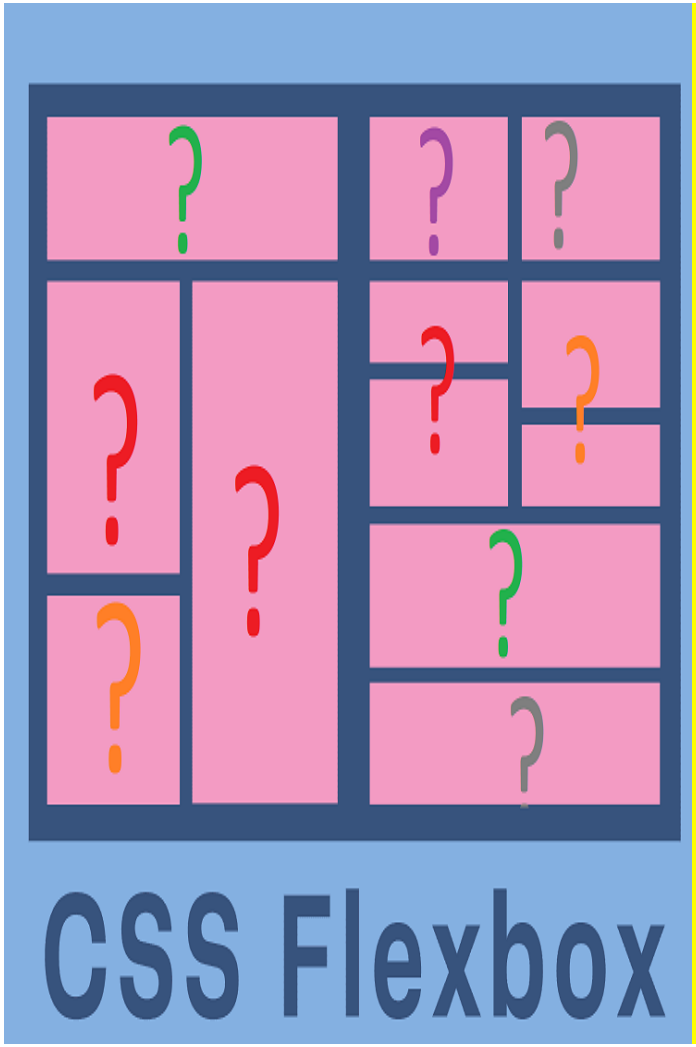
Flexbox



Por que usar o Flebox?

- **Distribuição de Espaço Automática**: Com Flexbox, é simples distribuir o espaço disponível entre elementos de maneira uniforme ou proporcional, dependendo das necessidades do layout.
- **Controle Individual de Itens**: Você pode controlar o tamanho, a ordem e o alinhamento de cada item individualmente, o que é útil para criar layouts altamente personalizados.

Flexbox



Por que usar o Flebox?

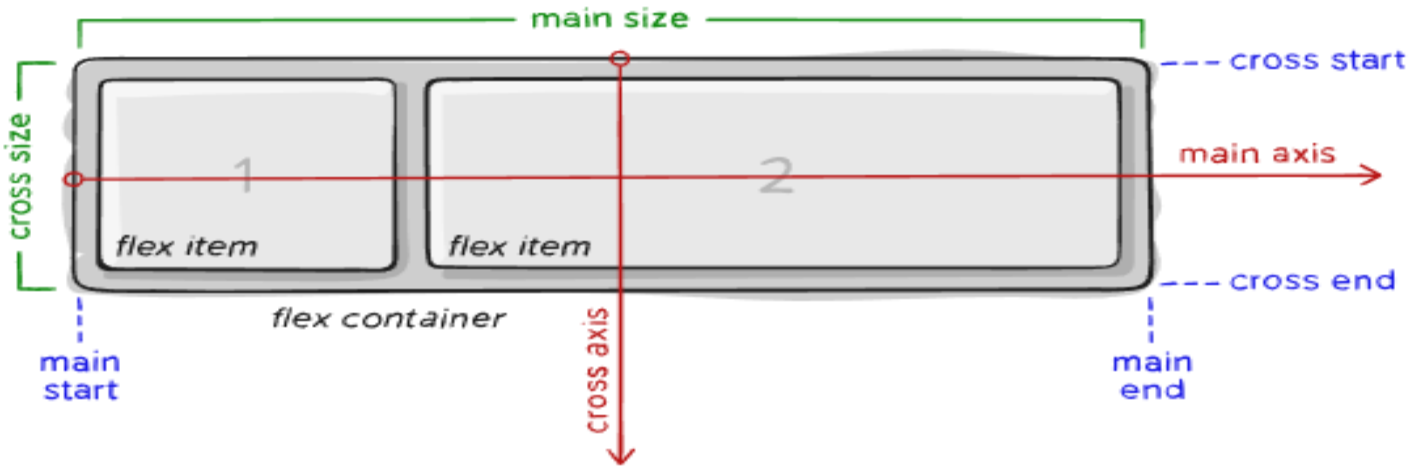
Evita Truques de Posicionamento: **Flexbox elimina a necessidade de usar truques de posicionamento complexos, como floats e clears, que podem ser propensos a erros e difíceis de manter.**

Código Mais Limpo: **Geralmente, o código CSS escrito usando Flexbox é mais legível e conciso.**

Suporte Amplo: **Flexbox é amplamente suportado pelos navegadores modernos.**

Flexbox

O flexbox possui um layout "regular" que é baseado em direções de fluxo em bloco e em linha, o layout flexível é baseado em "direções de fluxo flexível".

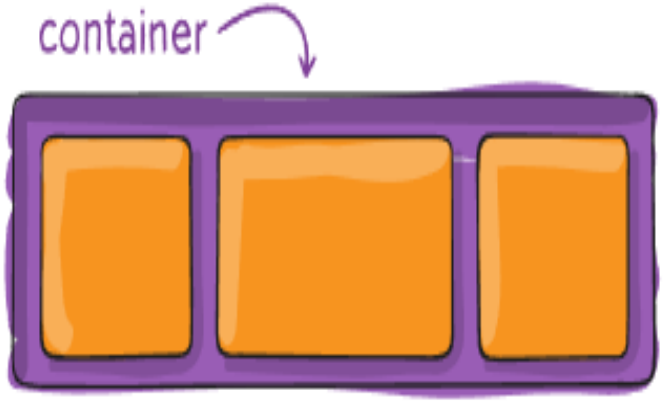


Os itens serão dispostos seguindo o eixo main axis(de main-start até main-end) ou o eixo transversal (de cross-start até cross-end).

Principais Propriedades do Flexbox

Propriedades do pai - (flex container)

É o elemento pai que envolve os itens flexíveis. Você aplica as propriedades do Flexbox a esse contêiner para controlar o comportamento dos itens filhos.



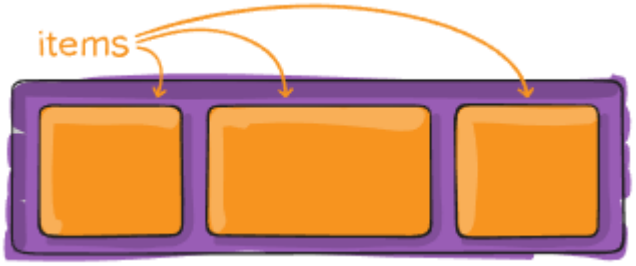

```css
.container {
  display: flex; /* or inline-flex */
}
```

Principais Propriedades do Flexbox

Propriedades dos filhos- (Flex Items)

São os elementos filhos dentro do contêiner flex. Você define como esses itens serão distribuídos, alinhados e dimensionados em relação ao contêiner pai.

Por padrão, os itens flexíveis são dispostos na ordem de origem. No entanto, a propriedade de ordem controla a ordem em que eles aparecem no flex container.

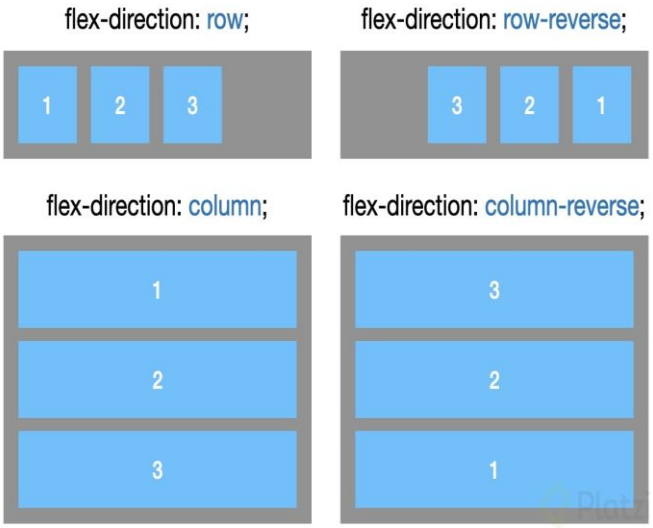



```
.item {
  order: 5; /* default is 0 */
}
```

Principais Propriedades do Flexbox

Flex-Direction

A propriedade flex-direction permite alterar a direção na qual os elementos flex serão exibidos ao longo do eixo principal.



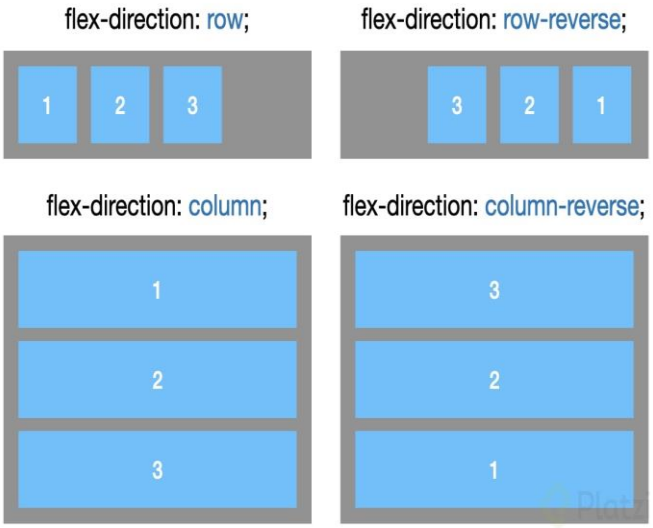

```
.container {
    display: flex;
    flex-direction: [row / row-reverse / column / column-reverse];
}
```

1. **row (padrão): Os itens são organizados em forma de linha da esquerda para a direita;**
2. **row-reverse: Os itens são organizados em forma exibição em linha da direita para a esquerda;**
3. **column: Os itens são organizados em forma de colunas iniciando de cima para baixo;**
4. **column-reverse: Os itens são organizados em forma de colunas iniciando de baixo para cima.**

Principais Propriedades do Flexbox

Flex-Direction

A propriedade flex-direction permite alterar a direção na qual os elementos flex serão exibidos ao longo do eixo principal.



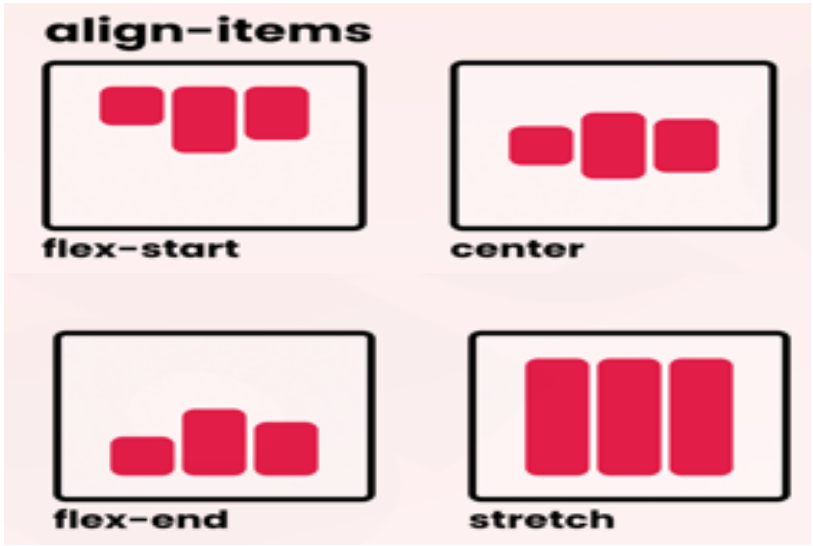

```
.container {
    display: flex;
    flex-direction: [row / row-reverse / column / column-reverse];
}
```

1. **row (padrão): Os itens são organizados em forma de linha da esquerda para a direita;**
2. **row-reverse: Os itens são organizados em forma exibição em linha da direita para a esquerda;**
3. **column: Os itens são organizados em forma de colunas iniciando de cima para baixo;**
4. **column-reverse: Os itens são organizados em forma de colunas iniciando de baixo para cima.**

Principais Propriedades do Flexbox

Align-items

A propriedade align-items define como as linhas são distribuídas ao longo do eixo transversal do container.



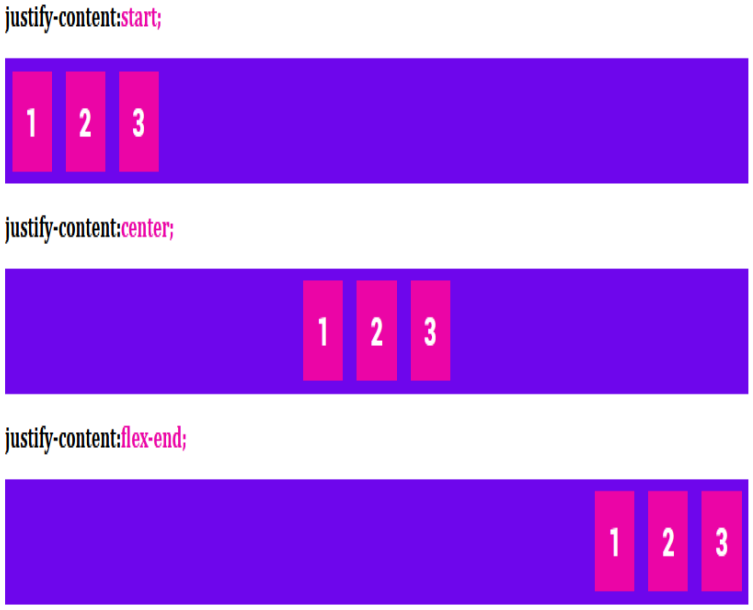

```
.container {
    display: flex;
    align-content: [stretch/flex-start/flex-end/center/space-between/space-around];
}
```

1. **stretch (padrão): estica os ítens para preencher o container, respeitando o min-width/max-width).**
2. **flex-start/ start / self-start: ítens são posicionados no início do eixo transversal. A diferença entre eles é sutil e diz respeito às regras de flex-direction ou writing-mode.**
3. **center: ítens são centralizados no eixo transversal.**
4. **baseline: ítens são alinhados de acordo com suas baselines.**

Principais Propriedades do Flexbox

justify-content

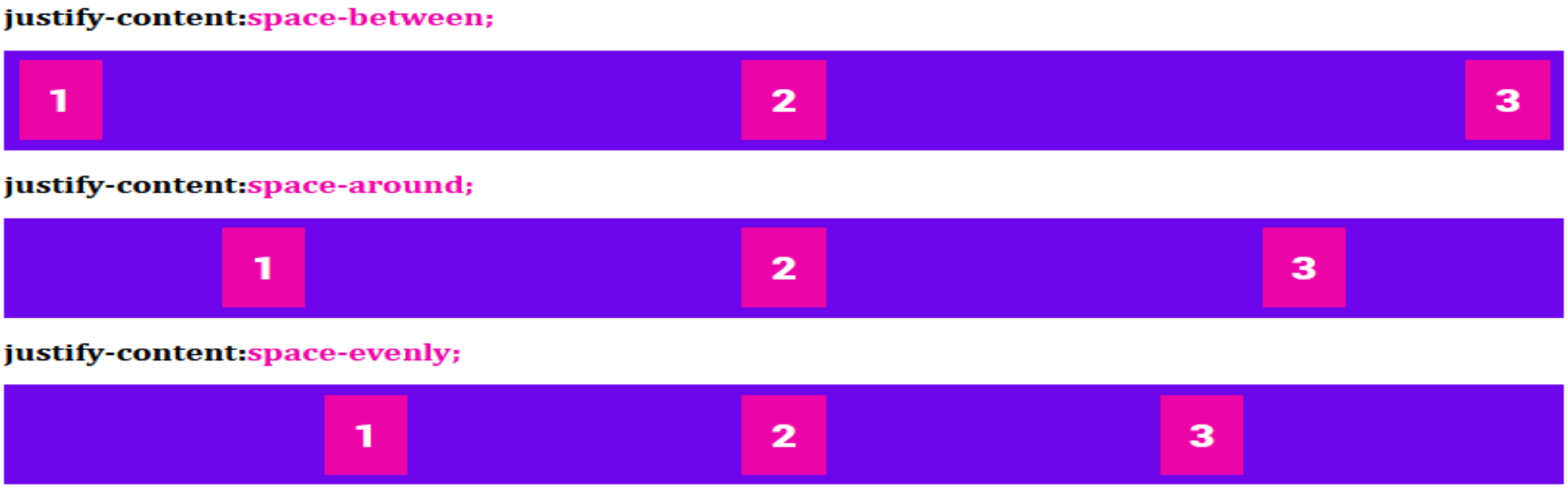
A propriedade justify-content é uma propriedade CSS que define como o navegador distribui o espaço entre e ao redor dos itens de conteúdo ao longo do eixo principal de um contêiner flexbox.



1. **flex-start (padrão): os itens são ordenados em direção ao início do eixo principal.**
2. **center: os itens são centralizados no eixo principal.**
3. **flex-end: os itens são compactados em direção ao final do eixo principal.**

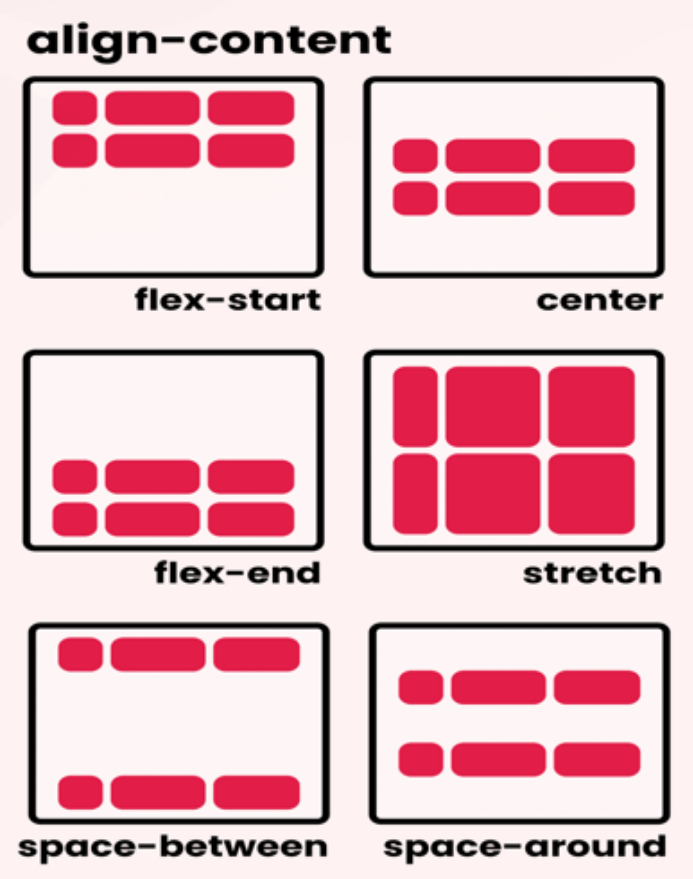
justify-content

4. **space-between**: os itens são distribuídos uniformemente ao longo do eixo principal, com o primeiro item no início do eixo principal e o último item no final do eixo principal.
5. **space-around**: os itens são distribuídos uniformemente ao redor do eixo principal, com espaço igual entre cada item e as bordas do contêiner flexbox.
6. **space-evenly**: os itens são distribuídos uniformemente ao longo do eixo principal, com espaço igual entre cada item.



Principais Propriedades do Flexbox

aling-content



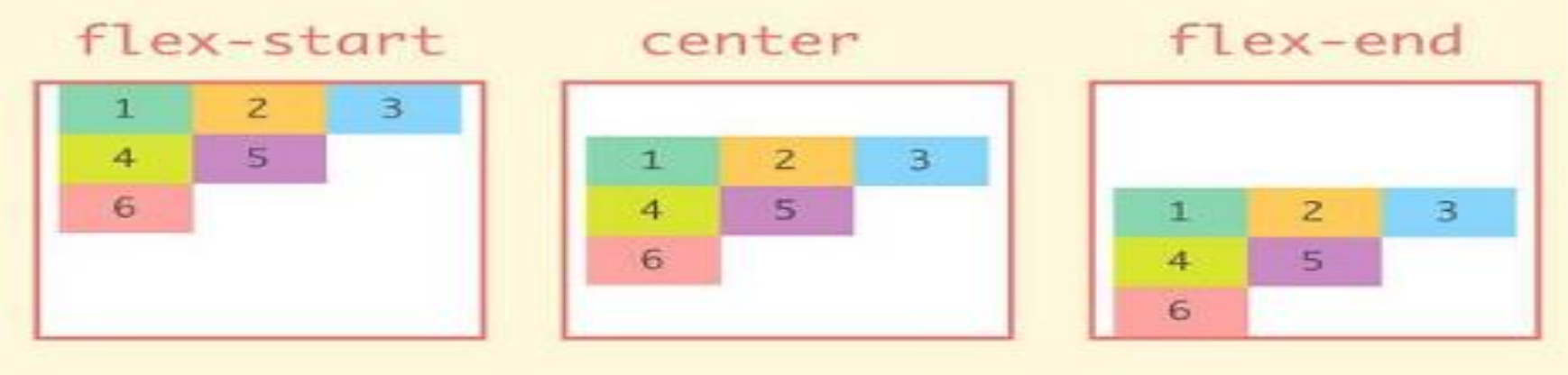
A propriedade **align-content** define como as linhas são distribuídas ao longo do eixo transversal do container.

Ela só terá efeito se o elemento tiver mais de uma linha, ou seja, se ele tiver elementos suficientes para quebrar a linha e a propriedade **flex-wrap: wrap** tiver sido definida.

Principais Propriedades do Flexbox

aling-content

1. **flex-start: As linhas são distribuídas a partir do início do eixo transversal;**
2. **center: As linhas são mantidas no centro do eixo transversal;**
3. **flex-end: As linhas são distribuídas a partir do fim do eixo transversal;**



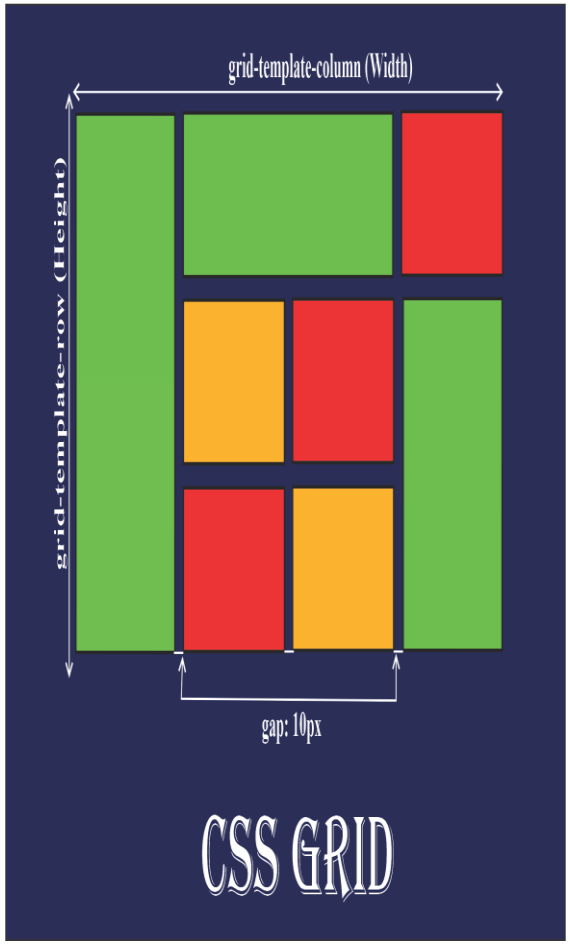
Principais Propriedades do Flexbox

aling-content

4. **space-between: A primeira linha é deslocada para o início do eixo transversal, a última é deslocada para o final do eixo transversal e as demais são distribuídas uniformemente entre eles;**
5. **space-around: As linhas são uniformemente distribuídas ao longo do eixo transversal. Aqui, porém, são atribuídas margens iguais à esquerda e à direita (ou acima e abaixo, dependendo da direção do eixo transversal).**
6. **stretch (padrão): As linhas são distribuídas uniformemente ao longo do eixo transversal, ocupando todo o espaço disponível;**

Grid Layout

Grid Layout



O que é Grid Layout?

O **Grid Layout** é uma técnica de layout em CSS que permite criar layouts de página de forma altamente flexível e controlada.

Ele organiza elementos HTML em uma **grade bidimensional**, o que significa que você pode **alinhar elementos em linhas e colunas**, criando um layout complexo e responsivo com facilidade.

Grid Layout - Conceitos-chave

- **Grid Container: É o elemento pai que contém os itens a serem organizados na grade. Para definir um elemento como um container de grade, você pode aplicar a propriedade CSS display: grid; a ele.**
- **Itens do Grid (Grid Items): São os elementos filhos do container de grade que serão organizados dentro da grade.**
- **Você pode especificar quais elementos são itens de grade aplicando a propriedade grid-column e grid-row a eles.**

Grid Layout - Conceitos-chave

- **Linhas e Colunas da Grade (Grid Rows and Columns):** O Grid Layout divide o espaço em linhas horizontais e colunas verticais. Você pode definir a estrutura da grade usando propriedades como grid-template-columns e grid-template-rows, e grid-template-áreas que permitem especificar o tamanho e o comportamento das linhas e colunas.
- **Espaçamento entre Células (Gaps):** Você pode definir espaçamentos entre as células da grade usando as propriedades grid-column-gap e grid-row-gap, ou a propriedade abreviada grid-gap.

Grid Layout - Conceitos-chave

- **Alinhamento de Conteúdo: O Grid Layout oferece controle preciso sobre o alinhamento de itens em relação às linhas e colunas da grade. Você pode usar propriedades como justify-items, align-items, justify-content e align-content para controlar o alinhamento horizontal e vertical dos itens.**
- **Posicionamento Personalizado (Grid Positioning): Além de organizar itens em linhas e colunas, o Grid Layout permite posicionar itens de forma personalizada em qualquer célula da grade, usando propriedades como grid-column-start, grid-column-end, grid-row-start e grid-row-end.**

Grid Layout

O Grid Layout é especialmente poderoso para criar layouts complexos e responsivos, e é uma alternativa eficaz ao sistema de posicionamento CSS mais antigo, como floats e posicionamento absoluto.

Ele facilita o design de páginas da web que se adaptam a diferentes tamanhos de tela e dispositivos.

Design Responsivo com CSS

Design Responsivo

Baseado na ideia de que o website deve ser bem exibido em todo tipo de tela, seja um desktop, um tablet ou um smartphone;

A técnica elimina a necessidade de ter duas versões do website, uma para smartphone (“mobile friendly”) e outra para desktops. Ao contrário disso, a ideia é utilizar o mesmo código HTML p/ as duas situações (não sendo necessário manter dois sites distintos);

Normalmente envolve a utilização de “media queries” da linguagem CSS em conjunto com outras propriedades e unidades relativas de tamanho;

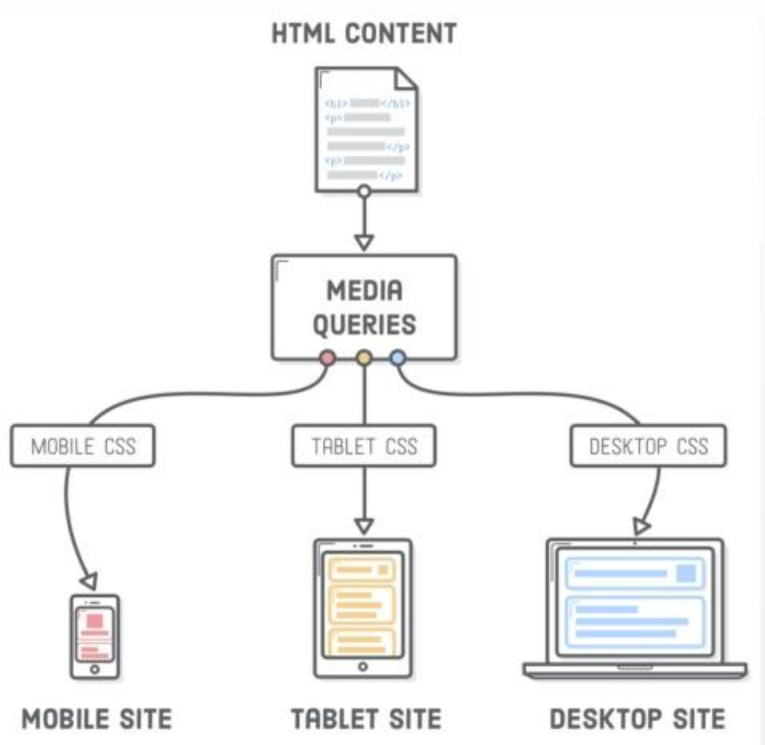
Design Responsivo

É a técnica de criar páginas para a web que se adaptem a diferentes resoluções, telas, dispositivos, sem precisar criar uma página específica para cada situação, mantendo sempre a acessibilidade e a usabilidade da página.

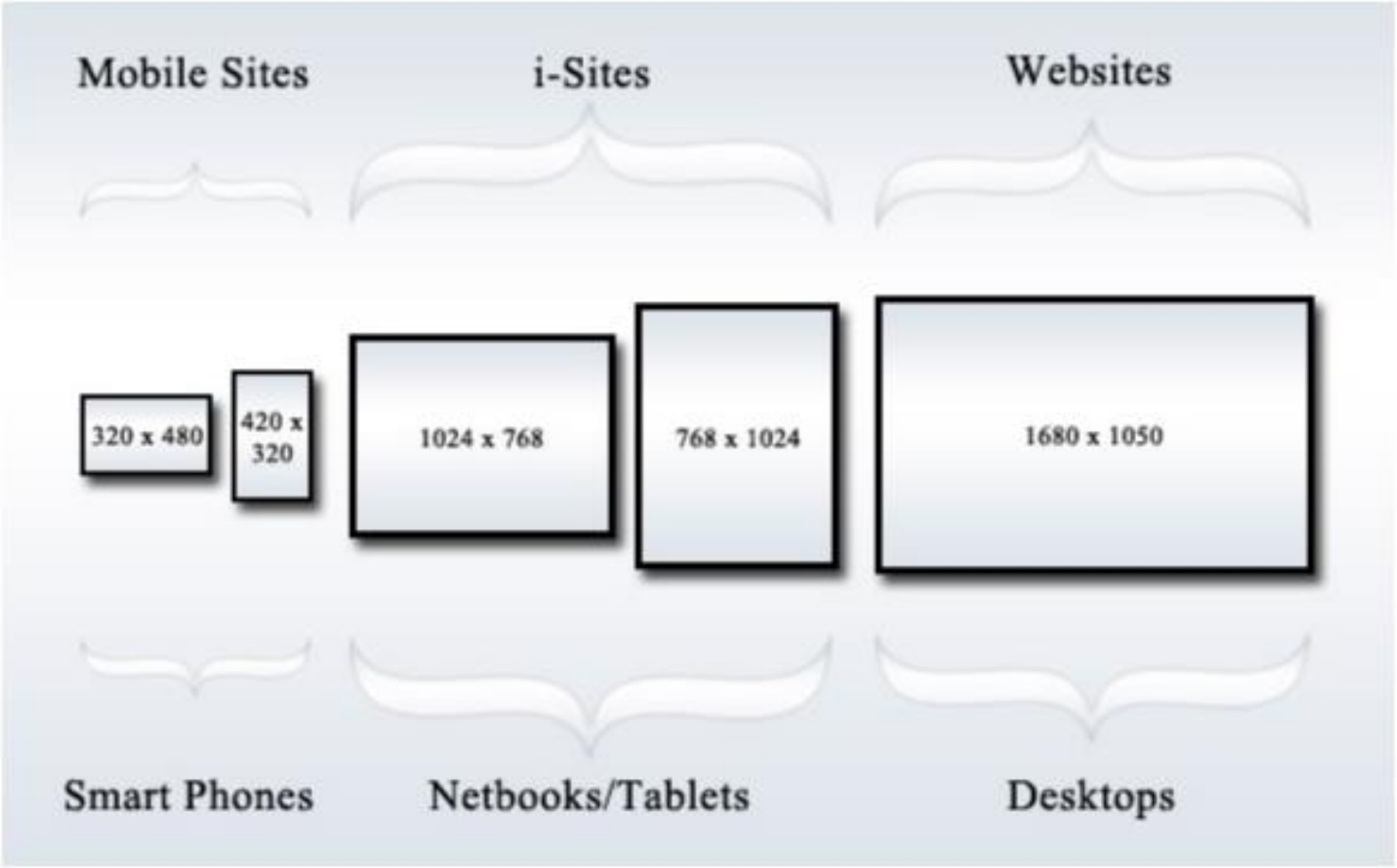
Media Queries

Permite inserir condições para a aplicação de estilos CSS;

Basicamente, diz ao navegador que ele deve aplicar ou ignorar um conjunto de estilos CSS dependendo do dispositivo do usuário (smartphone, tablet, desktop, etc.);



Media Queries



Media Queries – Principais resoluções utilizadas:

- 320 pixels - Smartphones no modo retrato.
- 480 pixels - Smartphones no modo paisagem.
- 600 pixels - Tablets pequenos.

Ex: Amazon Kindle (600×800)

- 768 pixels - Tablets maiores em modo retrato.

Ex: iPad (768×1024)

- 1024 pixels - Tablets maiores em modo paisagem, monitores antigos.
- 1200 pixels - Monitores wide

Media Queries

Especificam um estilo específico de acordo com a media, resolução, largura, etc. Os mais utilizados são:

- **all Para todos os dispositivos.**
- **handheld Para dispositivos de mão. Normalmente com telas pequenas e banda limitada.**
- **print Para impressão em papel.**
- **projection Para apresentações, como PowerPoint.**
- **screen Para monitores ou outros dispositivos com telas coloridas e com resolução adequada.**
- **tv Para dispositivos como televisores, ou seja, com baixa resolução, quantidade de cores e scroll limitado.**

Media Queries

Seu uso é feito dentro das tags <head> e </head>.

```
<link rel="stylesheet" href="estilo.css"
media="screen and (color)" />
```

- **Ou in-line:**

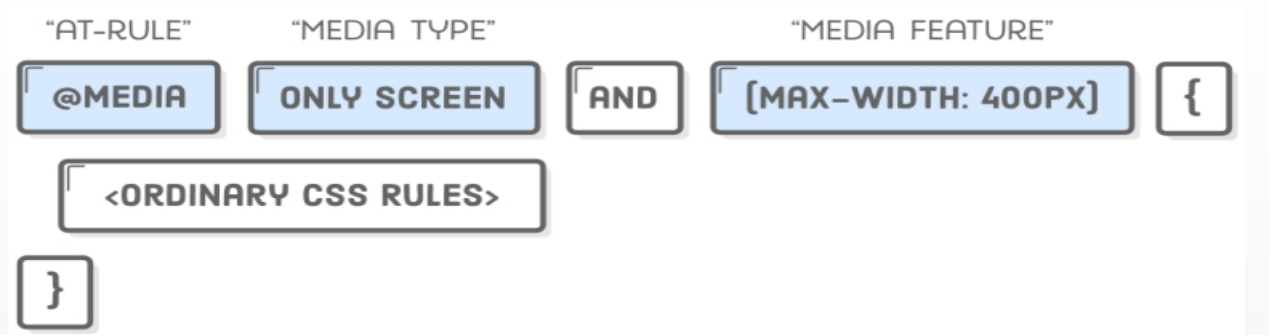
```
@media print {    /* estilos */ }
```

Media Queries

O navegador levará em consideração o código entre colchetes apenas quando as condições forem verdadeiras;

O valor only screen indica que os estilos devem ser aplicados apenas em dispositivos com telas (eles não serão aplicados, por exemplo, quando a janela estiver em modo de impressão – após pressionar Ctrl+P);

A opção max-width: 400px é denominada "media feature" e indica, neste exemplo, que as regras CSS devem ser aplicadas apenas em dispositivos com largura de tela igual ou inferior a 400 pixels;



Media Queries – Principais resoluções utilizadas:

```
<style>
          /* iPads (restrato e paisagem) */
   @media only screen and (min-device-width : 768px) and (max-device-width : 1024px) {

          /* estilos */

          body {
              width: 80%;
              margin: 0 auto;
               background-color: #EEE;
          }

           img {
              border-radius: 5px;
              margin-right: 10px;
              float: left;
           }
  }
</style>
```

Media Queries

Veja os exemplos CSS-Design-Responsivo-Ex02-A.html* ev CSS-Design-Responsivo-Ex02-B.html

Abra o exemplo A e diminua o tamanho da janela no navegador e observe a largura do painel central.

Repare que haverá um desperdício nas laterais quando a tela do dispositivo for bastante estreita, como a de um smartphone.

Isso acontece porque independentemente do dispositivo, fixou-se a largura do painel em 60% da página;

Abra o exemplo B, observe que o problema foi resolvido utilizando media queries.

Margens e Paddings Relativos

Utilizar margens e paddings com valores absolutos (px) pode prejudicar a exibição do website em dispositivos com telas pequenas;
Considere utilizar valores relativos, em percentuais. Exemplo:

```
.painel {
    padding: 10px 3%;
    background-color: white;
}
```

Abra o arquivo CSS-Design-Responsivo-Ex02-C.html no navegador e reduza a largura da janela.

Observe o comportamento do padding nas laterais do texto.

Media Queries

Trabalho: Fazer um resumo do assunto:

Usando Media Queries

Disponível no link :

https://developer.mozilla.org/pt-BR/docs/Web/Guide/CSS/CSS_Media_queries

Media Queries

Há muitas outras opções que podem ser utilizadas com media queries.

Para mais exemplos, acesse:

https://developer.mozilla.org/en-US/docs/Web/CSS/@media

Referências

- www.w3schools.com/html/html_tables.asp
- www.w3schools.com/html/html_forms.asp
- www.w3.org/Style/CSS/
- www.w3.org/Style/Examples/011/firstcss
- www.w3.org/Style/LieBos2e/enter/
- www.w3.org/MarkUp/Guide/Style
- www.w3.org/Style/CSS/learning
- internetingishard.com/html-and-css/responsive-design/

Obrigado