Desenvolvimento Web

Aula 7

Prof. Calebe Conceição

O que vimos na última aula?

Fizemos práticas usando HTML e CSS.

Revisamos a estrutura do DOM e como selecionamos seus elementos para estilizar

Exercitamos o uso do ambientes de desenvolvimento VSCODE

Aprendemos a colocar nossa página para acesso público na rede local

Customizamos uma versão de página pessoal

O que vamos ver hoje:

Aprofundaremos um pouco mais os conceitos de CSS

Conheceremos novas estruturas de estilização de layout

Praticaremos sobre nosso site de referência

Display

Para começar a aula..

- Para começar a aula
  - Criem um documento HTML
  - Criem um documento CSS
  - Criem no HTML o link para o arquivo CSS (<link … >)

Display

- O que é display?

Display

- é o atributo que guarda a forma de mostrar o elemento html
- já vimos no html a ideia do display
  - inline
  - block
  - inline-block
  - none
  - (tem mais, mas não são tão comuns…)

Display

- Algumas TAGs HTML têm display padrão específico.
  - TAGs que são *block* por padrão
    - Estruturais: , <section>, <article>, <header>, <footer>, <main>, <aside>
    - Textuais (de bloco): <p>, <h1> a <h6>, <blockquote>, <pre>
    - Listas: <ul>, <ol>,
    - Outros: <form>, <table>, <address>, <hr>
  - TAGs que são *inline* por padrão
    - Textuais , <a>, , , <b>, <i>, <u>, <abbr>, <cite>
    - Inline de formulário <label>, <input>, <select>, <textarea>, <button> (alguns variam ligeiramente)
- Via de regra, podemos modificar o display padrão desses elementos por meio do CSS.

Inline

- Mantém os elementos em linha, ocupando só o que precisa.

Inline

- Vamos praticar, repita o que o tutor está fazendo.
  - **Coloquem um div com um p dentro no html com algum texto dentro. Escreva no texto apenas seu nome.**
  - **Coloquem o estilo ao lado no css**

```
p, div {
    border: solid 2px black;
}
```

Inline

```
p, div {
    border: solid 2px black;
}
```

- O que aconteceu aí?
  - Ocuparam toda a linha, porque são TAGs block por padrão

Inline

- O que aconteceu aí?
  - Agora alterem o estilo para colocar os dois elementos como inline
  - **O que aconteceu aí?**

```
p, div {
    border: solid 2px black;
    display: inline;
}
```

Inline

- A declaração do atributo display como inline faz com que um elemento que era diferente de inline vire inline
- Lembrando: Inline só ocupa o espaço necessários para mostrar o conteúdo do elemento

```
p, div {
    border: solid 2px black;
    display: inline;
}
```

Block

- Qual elemento do html que é block?

Block

- Qual elemento do html que é block?
  - **Coloquem uma TAG span (#span1) e dentro dele outra TAG span (#span2) com um texto dentro de cada span**
  - **Qual o efeito? Coloquem o estilo ao lado no css**

```
#span1, #span2 {
    border: solid 2px green;
}
```

Block

- O que aconteceu aí?
  - alterem o estilo para colocar os dois spans como block
  - **O que aconteceu aí?**

```
#span1, #span2 {
    border: solid 2px green;
    display: block;
}
```

Block

- **A declaração do atributo display como block faz com que um elemento que era diferente de block vire block**
- block ocupa a linha inteira
  - dá quebra de linha depois

```
#span1, #span2 {
    border: solid 2px green;
    display: block;
}
```

Inline-block

- **Qual elemento do html que é inline-block?**

Inline-block

- Qual elemento do html que é inline-block?
  - **Tem poucos**
    - button
    - select
    - img
    - input

Inline-block

- Voltem para o exemplo dos spans
  - alterem o estilo dos spans para o que está aqui ao lado
  - o que aconteceu?

```
#span1, #span2 {
  border: solid 2px green;
  width: 500px;
  height: 500px;
}
```

Inline-block

- Espero que não tenha acontecido nada
- Pois então…
  - não faz diferença definir width e height de elementos inline
  - os elementos continuam sendo limitados pelo conteúdo

```
#span1, #span2 {
  border: solid 2px green;
  width: 500px;
  height: 500px;
}
```

Inline-block

- Com o inline-block da para definir o tamanho e largura
- Teste o código aí
- **A diferença de um elemento inline para inline-block é que em um inline-block dá para definir a altura e largura dele.**

```
#span1, #span2 {
    border: solid 2px green;
    width: 500px;
    height: 500px;
    display: inline-block;
}
```

None

- **O que vocês acham que faz o display: none?**

None

- Se display é mostrar
  - display: none
    - é **não mostrar**! :-)
- Atualizem o estilo dos spans com o que está ao lado
  - **O que aconteceu?**

```
#span1, #span2 {
    border: solid 2px green;
    width: 500px;
    height: 500px;
    display: none;
}
```

None

- Sumiu com os elementos que estão com o estilo de **display:none**;
- Como se tivesse tirado do html o elemento
  - Isso é bastante usado no javascript, para realizar efeitos visuais

```
#span1, #span2 {
    border: solid 2px green;
    width: 500px;
    height: 500px;
    display: none;
}
```

Visibility

```
<img id="img1" src="imgs/bg1.jpg"/>
<img id="img2" src="imgs/bg2.jpg"/>
<img id="img3" src="imgs/bg3.jpg"/>
```

- Pesquisem no google imagens de gatinho
- Salve-as como bg1.jpg, bg2.jpg e bg3.jpg
- Coloque as no html de vocês com as TAGs ao lado

Visibility

- Coloquem o estilo ao lado no CSS de vocês

```
img {
  width: 30% ;
  margin: 1%;
}
```

Visibility

- Para exemplificar o visibility
- Coloque agora o estilo dos ids das imagens

```
img {
  width: 30% ;
  margin: 1%;
}
#img1 {
  display: none;
}
#img2 {
  visibility: hidden;
}
```

Visibility

- **Qual a diferença entre display:none e visibility:hidden?**

```
img {
  width: 30% ;
  margin: 1%;
}
#img1 {
  display: none;
}
#img2 {
  visibility: hidden;
}
```

Visibility

- **display:none**
  - fica como se o elemento não estivesse no html mesmo
- **visibility:hidden**
  - só esconde o elemento
  - mas deixa a posição dele “guardada"

```
img {
  width: 30% ;
  margin: 1%;
}
#img1 {
  display: none;
}
#img2 {
  visibility: hidden;
}
```

Position

Para começar a aula..

- Para começar a aula
  - Criem um documento HTML
  - Criem um documento CSS
  - Criem no HTML o link para o arquivo CSS (<link … >)

Position

- O que é position?
  - Sim, eu sei que position é posição...

Position

- O que é position?
  - é como o elemento HTML vai ser colocado dentro da página HTML
  - dependendo do position
    - vai ser colocado em um lugar diferente da página

Position

- Existem 5 tipos de posições
  - static (padrão)
  - relative
  - fixed
  - absolute
  - sticky

Static

- Criem um div no html
  - Coloquem algum texto para aparecer na tela
- **Coloquem no CSS de vocês o estilo ao lado**

```
div {
    position: static;
    width: 250px;
   border: 4px solid red;
   padding: 15px;
}
```

Static

- Fica com a posição padrão do html
  - Ou seja, conforme a gente for colocando os elementos html
  - Eles vão sendo exibidos um depois do outro

```
div {
    position: static;
    width: 250px;
    border: 4px solid red;
    padding: 15px;
}
```

Static

- Fica com a posição padrão do html
  - Não dá para definir o atributos top, right, bottom, left
  - Até dá, mas não faz diferença….

```
div {
    position: static;
    width: 250px;
    border: 4px solid red;
    padding: 15px;
}
```

Relative

- Criem um div no html
  - Coloquem algum texto para aparecer na tela
- **Coloquem no CSS de vocês o estilo**

```
div {
    position: relative;
    left: 30px;
    top: 30px;
    width: 250px;
   border: 4px solid red;
   padding: 15px;
}
```

Relative

- Fica deslocado de acordo com a posição normal dele
  - Definir com os atributos top, right, bottom, left

```
div {
    position: relative;
    left: 30px;
    top: 30px;
    width: 250px;
   border: 4px solid red;
   padding: 15px;
}
```

Relative

- Se colocar um span dentro do div
  - o estilo do lado
  - fica deslocado dentro do div
    - que é deslocado em relação a posição esperada dele
  - ou seja, há um acúmulo de deslocamentos

```
div {
   position: relative;
   left: 30px;
   top: 30px;
   width: 250px;
  border: 4px solid red;
  padding: 15px;
}

span {
   position: relative;
   left: 30px;
   top: 30px;
}
```

Relative

- Cuidar com o relative porque ele desloca o elemento em relação a posição que ele deveria ocupar
  - teste relative
  - teste não relative

```
.relative {
    position: relative;
    left: 30px;
    top: 30px;
    width: 250px;
   border: 4px solid red;
   padding: 15px;
}
```

Fixed

- Fixed fica fixo na tela
- Independente de o que acontecer
- A posição é em relação ao valores de right, bottom, left, top definidos

```
.relative {
   position: fixed;
   right: 0;
   bottom: 0;
   width: 250px;
  border: 4px solid red;
  padding: 15px;
}

p {
   height: 2000px;
}
```

Fixed

- Troquem os estilos para fixed
- Coloquem um p aí com algum texto também

```
.relative {
   position: fixed;
   right: 0;
   bottom: 0;
   width: 250px;
  border: 4px solid red;
  padding: 15px;
}

p {
   height: 2000px;
}
```

Absolute

- Coloquem os dois estilos de absolute aí
- Criem duas divs diferentes com as classes
  - absolute1
  - absolute2

```
.absolute1 {
   position: absolute;
  top: 40px;
  left: 40px;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
.absolute2 {
  position: absolute;
  top: 20px;
  left: 20px;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

Absolute

- Coloquem uma dentro da outra

```
.absolute1 {
   position: absolute;
  top: 40px;
  left: 40px;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
.absolute2 {
  position: absolute;
  top: 20px;
  left: 20px;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

Absolute

- A ideia do **absolute** é que ele é **absolute** em relação ao elemento pai **absolute** dele
- Se tiver irmão, ele não se importa...

```
<div class="absolute1">
  Teste de posição.
  <div class="nao-absolute">
    Não absolute...
  </div>
  <div class="absolute2">
    Teste de posição.
  </div>
</div>
```

Sticky

- Troquem os estilos para sticky
- O que acontece?

```
div {
   position: sticky;
   top: 0;
   width: 250px;
  border: 4px solid red;
  padding: 15px;
}

p {
   height: 2000px;
}
```

Sticky

- Acompanha a rolagem da tela
  - Mas ele volta para posição inicial quando na parte inicial
- Tem de colocar mais elementos na página para aparecer...

```
.relative {
    position: fixed;
    right: 0;
    bottom: 0;
    width: 250px;
   border: 4px solid red;
   padding: 15px;
}

p {
    height: 500px;
}
```

Responsividade

Responsividade

- Alguém sabe do que se trata?

Responsividade

- Alguém sabe do que se trata?
  - são aqueles que adaptam o tamanho das suas páginas (alteração do layout) ao tamanho das telas que estão sendo exibidos
    - como as telas de celulares e tablets.

Responsividade

- Existem várias formas de construir isso
  - CSS Media Query
  - Flexbox
  - Frameworks próprios de linguagem
  - ...

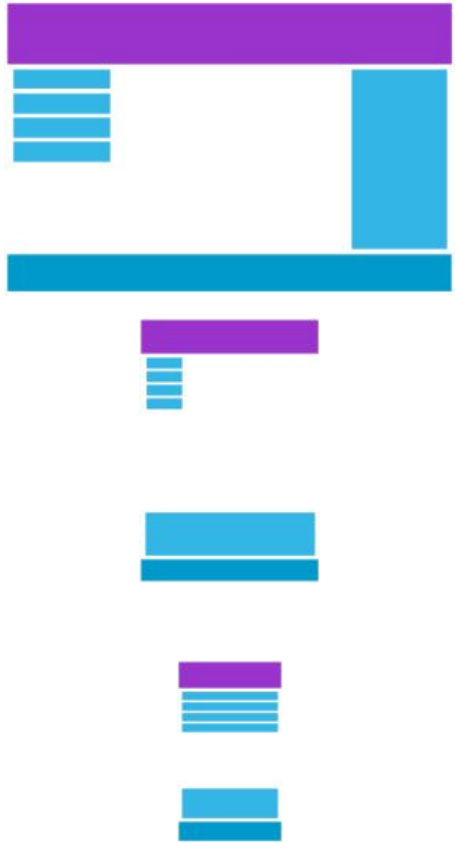
Media Query

Media Query

- Alguém sabe do que se trata?
  - Com base no nome, consegue chutar?

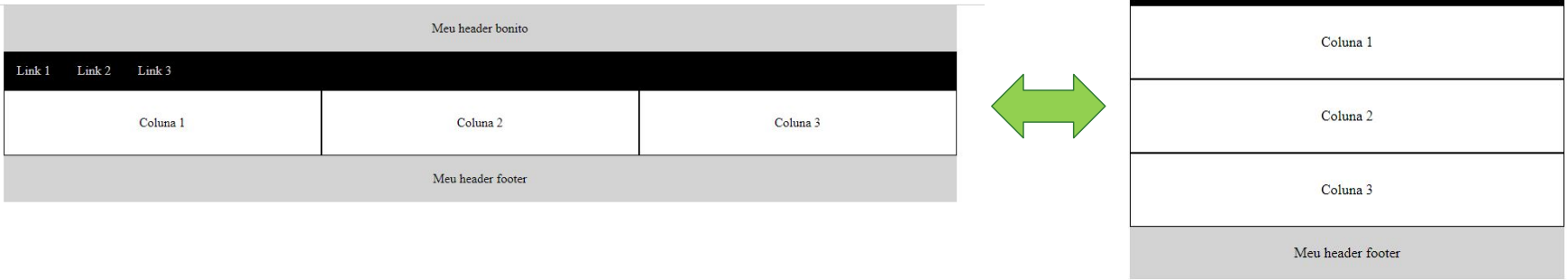
Media Query

- Alguém sabe do que se trata?
  - Os media types definem para que tipo de media um certo código CSS é direcionado.
  - Pode ser interpretado por qualquer tipo de dispositivo
  - Exemplo: abra o link.



Media Query

- Para demonstrar isso vamos considerar um Layout que fizemos
- Baixe os códigos pelo link



Media Query

- Media Query
  - Introduzido no CSS3
  - Permite especificar filtros de estilo para uma página HTML
  - Utiliza a sintaxe abaixo

```
@media [not|only] [media type] and ([query]) {
       estilos
}
```

Media Query

```
@media [not|only] [media type] and ([query]) {
        estilos
}
```

- @media
  - Define que dentro de um CSS, um filtro vai ser definido
  - É obrigatório

```
@media
```

Media Query

```
@media [not|only] [media type] and ([query]) {
        estilos
}
```

- not ou only
  - É opcional
  - Se usado o **not**
    - Faz com que a configuração selecionada não seja executada
  - Se usado o **only**
    - Executa somente na configuração selecionada. É o padrão, não necessita ser adicionado.

```
@media only
```

Media Query

```
@media [not|only] [media type] and ([query]) {
        estilos
}
```

- Media Type
  - Define o tipo de tela que vai valer:
    - **All**: Para todos os dispositivos.
    - **Braille**: Para dispositivos táteis.
    - **Print**: Para impressão em papel.
    - **Screen**: Para monitores ou dispositivos com telas coloridas e resolução adequada.
    - Existem muitos outros ...

```
@media only screen
```

Media Query

```
@media [not|only] [media type] and ([query]) {
        estilos
}
```

- Query
  - Permite definir filtros
  - Podem ter vários divididos por **and** ou **or**

```
@media only screen and (min-width: 600px) and (max-width: 900px) {
}
```

Media Query

```
@media [not|only] [media type] and ([query]) {
        estilos
}
```

- Query
  - width: largura (min e max também)
  - height: altura (min e max também)
  - orientation: Orientação (vertical e horizontal/landscape)
  - color
  - resolution
  - muitos outros (recomendo olhar a documentação)

```
@media only screen and (min-width: 600px) and (max-width: 900px) {
}
```

Media Query

```
@media [not|only] [media type] and ([query]) {
        estilos
}
```

- Query
  - **width: largura (min e max também)**
  - **height: altura (min e max também)**
  - orientation: Orientação (vertical e horizontal/landscape)
  - color
  - resolution
  - muitos outros (recomendo olhar a documentação)

Os mais relevantes

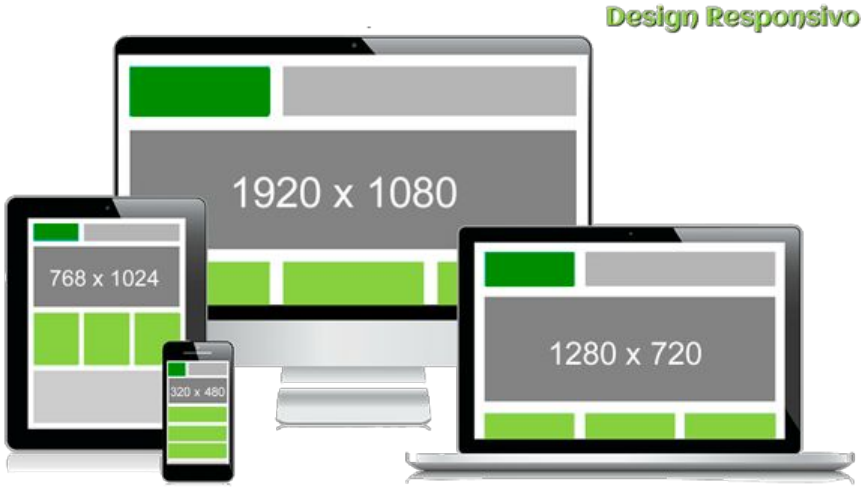
```
@media only screen and (min-width: 600px) and (max-width: 900px) {
}
```

Media Query

- Query
  - **width** e **height**
    - Quando utilizar?

Media Query

- Query
  - **width** e **height**
    - Quando utilizar?
    - Usamos a largura de referência dos dispositivos em **PX**
    - Para não precisar considerar todos os tamanhos consideramos **breakpoints**
      - **Eles são arbitrários**



Media Query

- Breakpoints (width) mais comuns:
  - 480px: Extra small
  - 768px: Small screen
  - 992px: Medium screen
  - 1200px: Large screen
  - 1600px Extra Large screen
  - 1900px Extra Large screen

Media Query

- Bora testar
  - Copie o código do lado para eu CSS
  - Adicione ele lá, não apague o anterior

```
/* Telas grandes */
@media only screen and (min-width: 1200px) {
    body {
        width: 1200px;
        margin: 0 auto;
    }
}
```

Media Query

- Bora testar
  - Copie o código do lado para eu CSS
  - Adicione ele lá, não apague o anterior
  - **Alterem o tamanho da janela**
    - **O que aconteceu?**

```
/* Telas grandes */
@media only screen and (min-width: 1200px) {
    body {
        width: 1200px;
        margin: 0 auto;
    }
}
```

Media Query

- Bora testar
  - Copie o código do lado para eu CSS
  - Adicione ele lá, não apague o anterior
  - **Alterem o tamanho da janela**
    - **O que aconteceu?**

```
/* Telas médias */
@media only screen and (min-width: 768px)
and (max-width: 1199px){
    body {
       width: 768px;
         margin: 0 auto;
    }
    .coluna {
        padding: 27px;
        width: 26%;
    }
}
```

Media Query

- Bora testar
  - Copie o código do lado para eu CSS
  - Adicione ele lá, não apague o anterior
  - **Alterem o tamanho da janela**
    - **O que aconteceu?**

```
/* Telas de celular pequenas */
@media only screen and (max-width: 767px){
    body {
        width: 100%;
    }
    .coluna {
        width: 100%;
        float: none;
    }
}
```

Media Query

- A maioria dos sites usa media query
  - Abram algum e usem o inspecionar para ver que regras eles usam

FlexBox

Flex Box

- CSS2 suportava somente Float Box Layouts
  - Alguns problemas conhecidos:
    - Dificuldade de conter elementos
    - Dependente da ordem no código fonte
    - Dificuldade com colunas com pesos iguais
    - Não permitia “float: center”
    - Não permitia centralizar na vertical

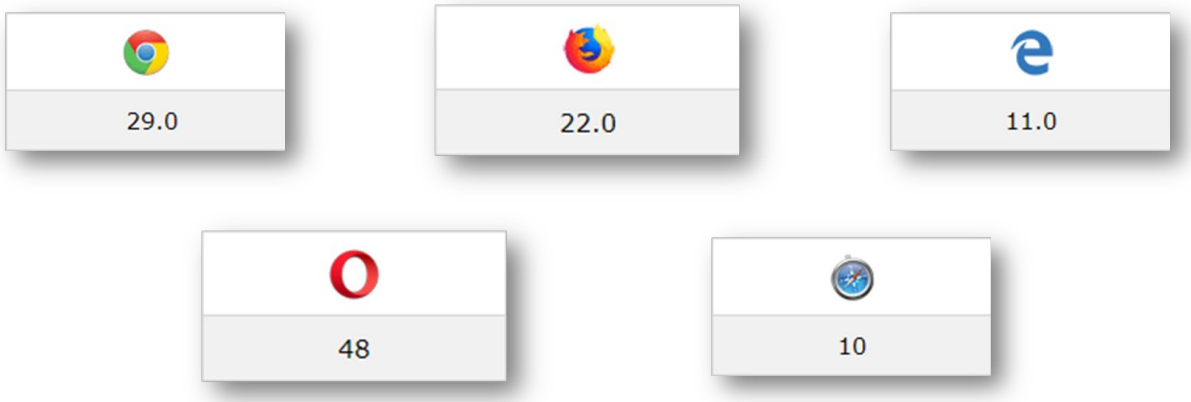
Flex Box

- Modelo **Flexible Box Layout**, também conhecido como **CSS Flexbox**, é um recurso disponibilizado pelo CSS3 para construção de páginas de layout simples de uma forma mais flexível
  - Especificação: https://www.w3.org/TR/css-flexbox-1/
  - Primeira release estável: 2012

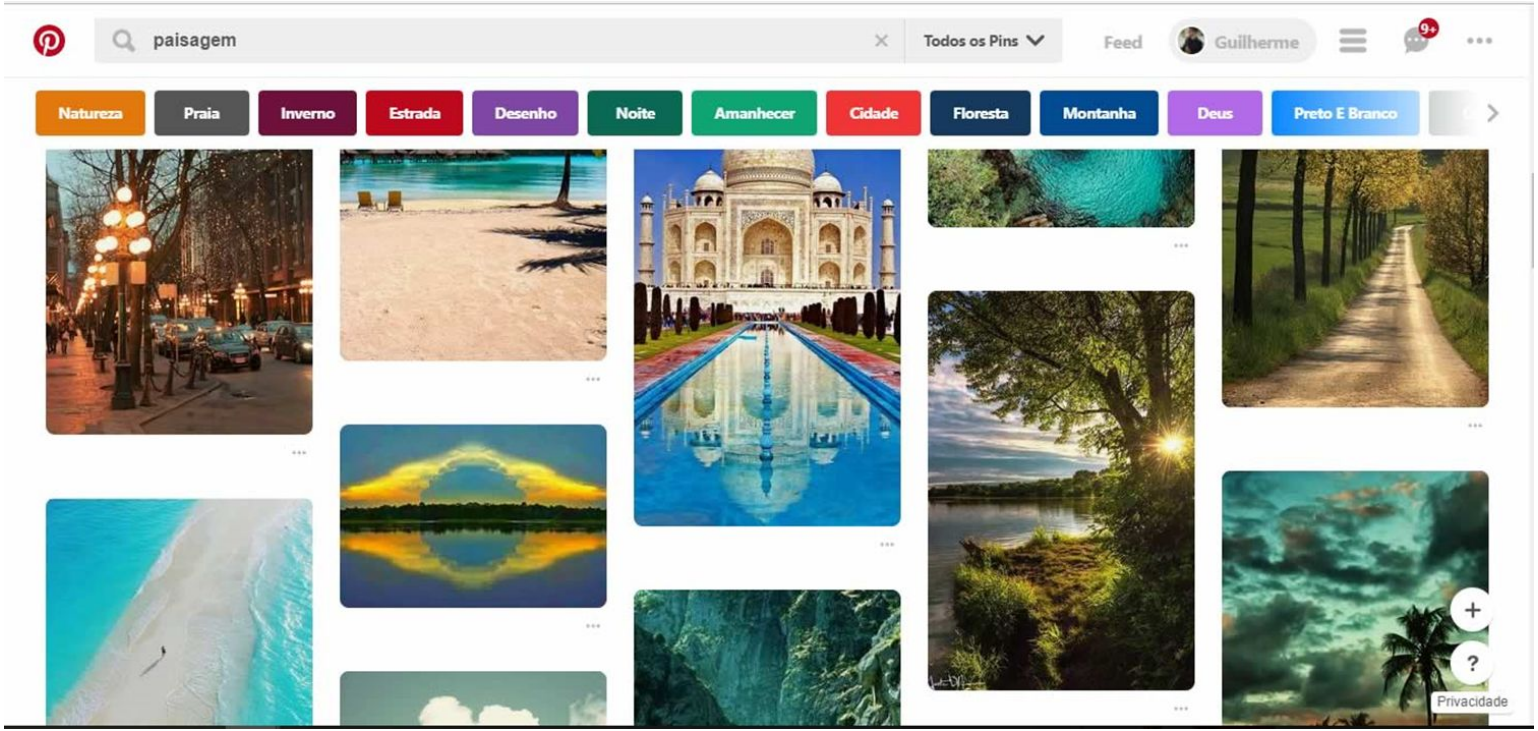
  - O Flexbox facilita a construção de layouts responsivos sem a necessidade de atribuir explicitamente **posicionamento** e **float**

Flex Box

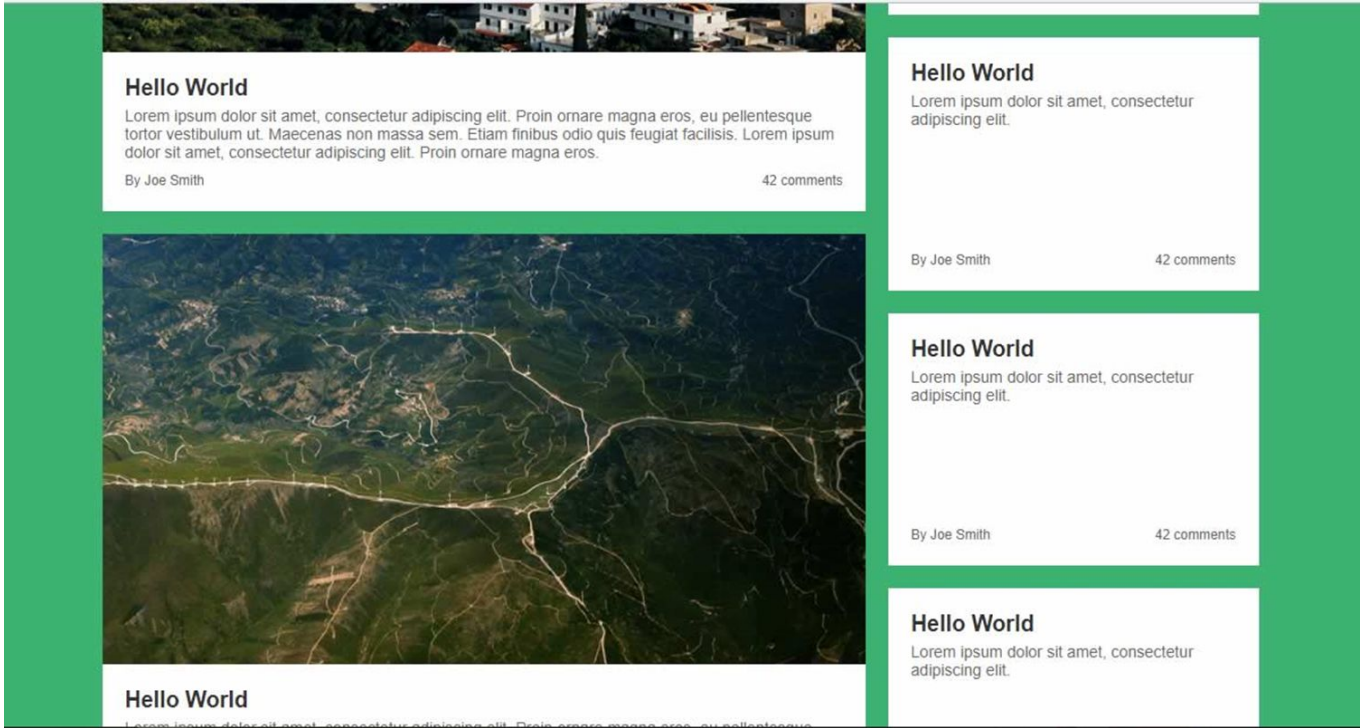
- Flexbox
  - Atualmente suportável por todos os navegadores



Exemplo de Página: Pintrest



Exemplo de página template

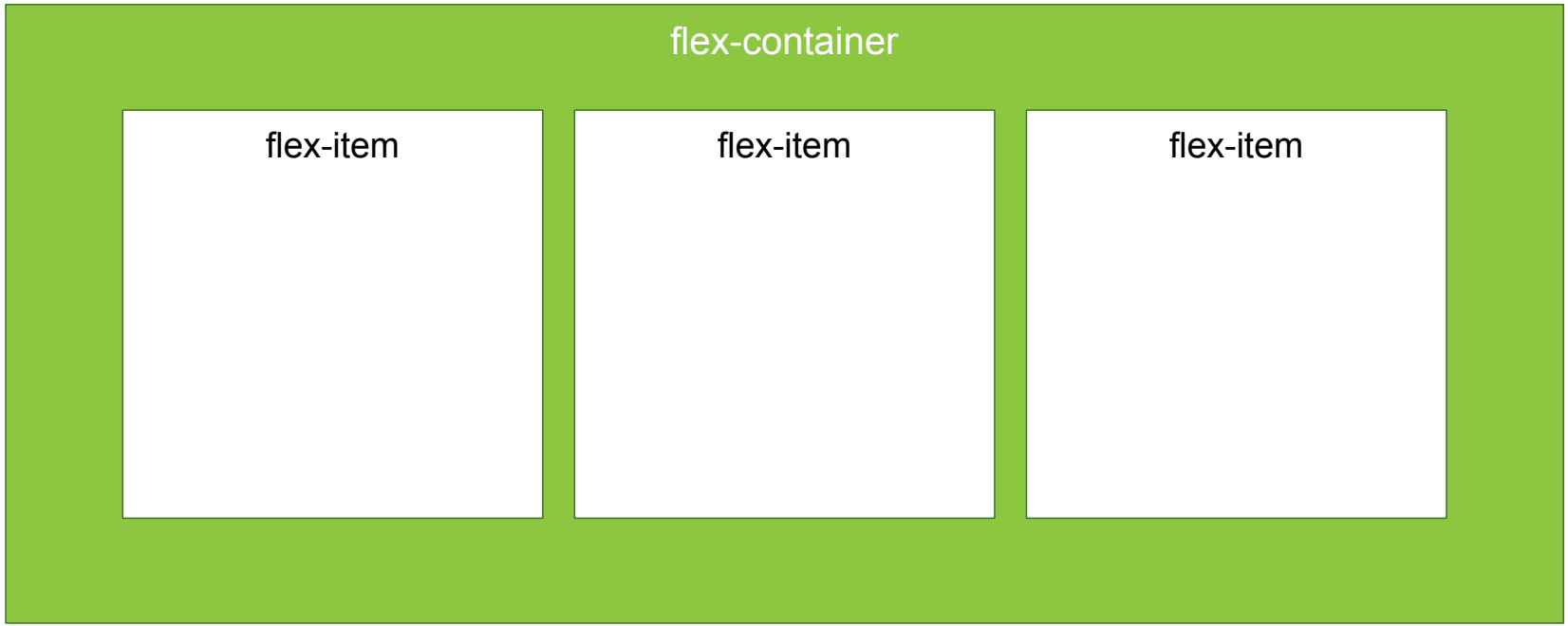


Flex Box

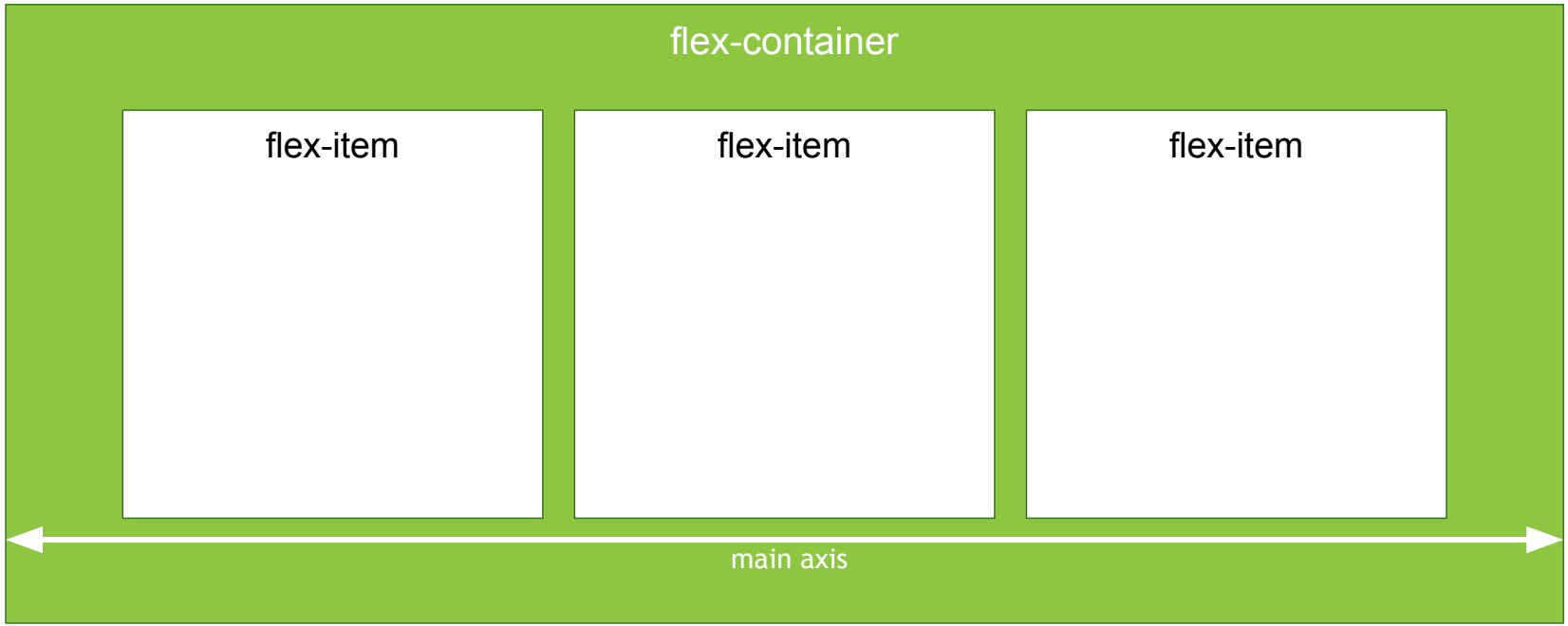
- Como ele funciona?
  - Ele é dividido em duas partes:
    - flex-container
    - flex-item

Estrutura do Flexbox

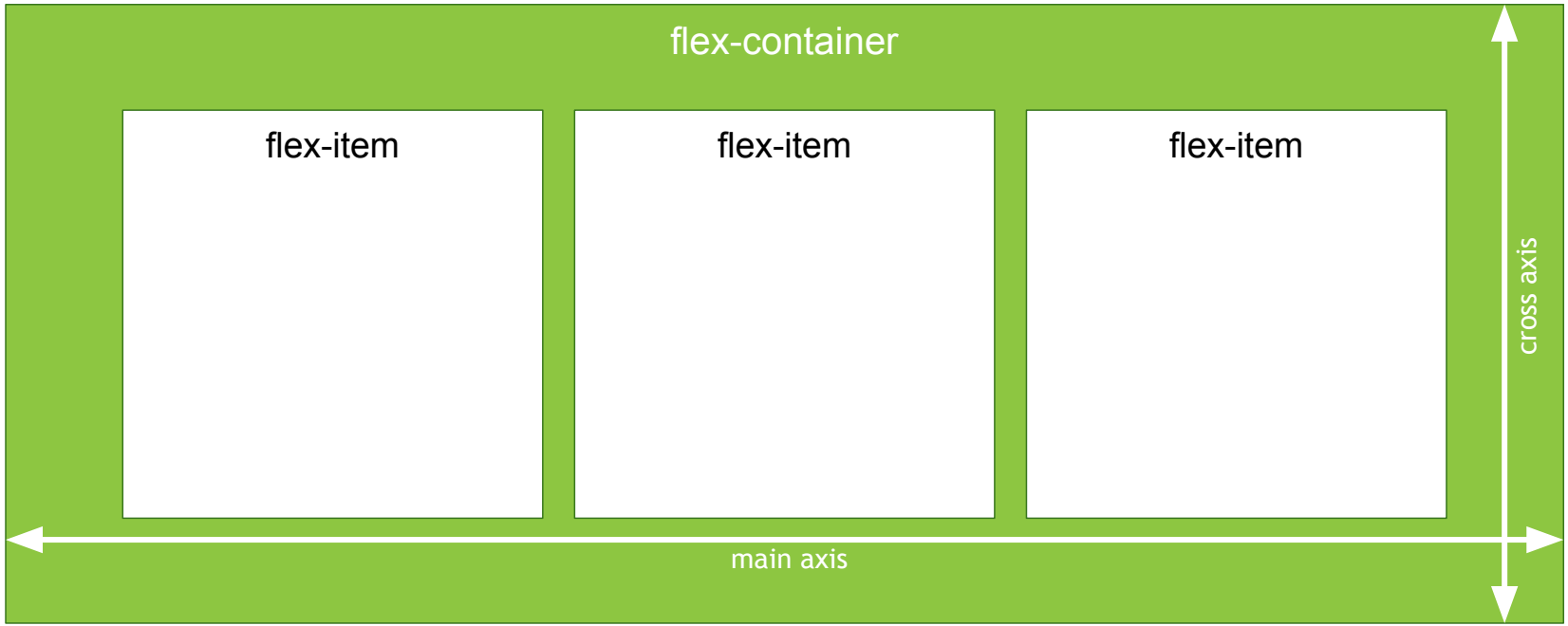
Estrutura do Flexbox



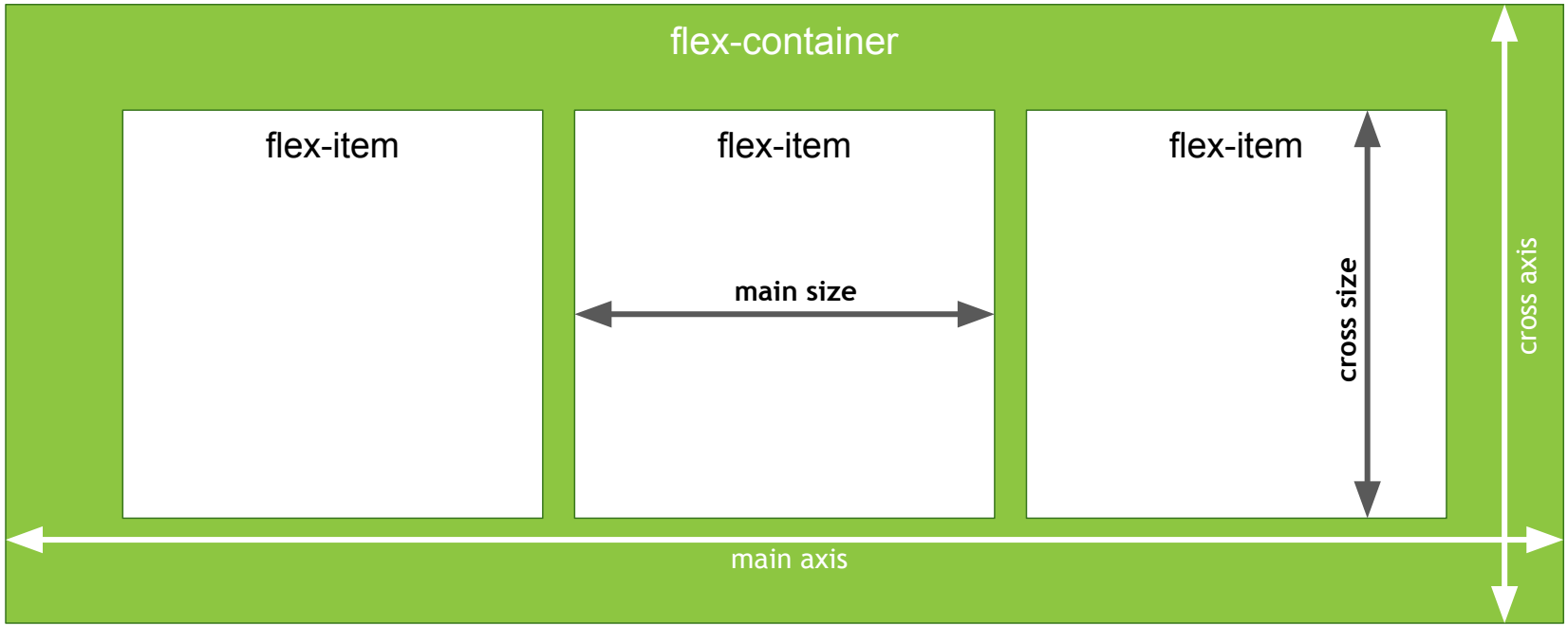
Estrutura do Flexbox



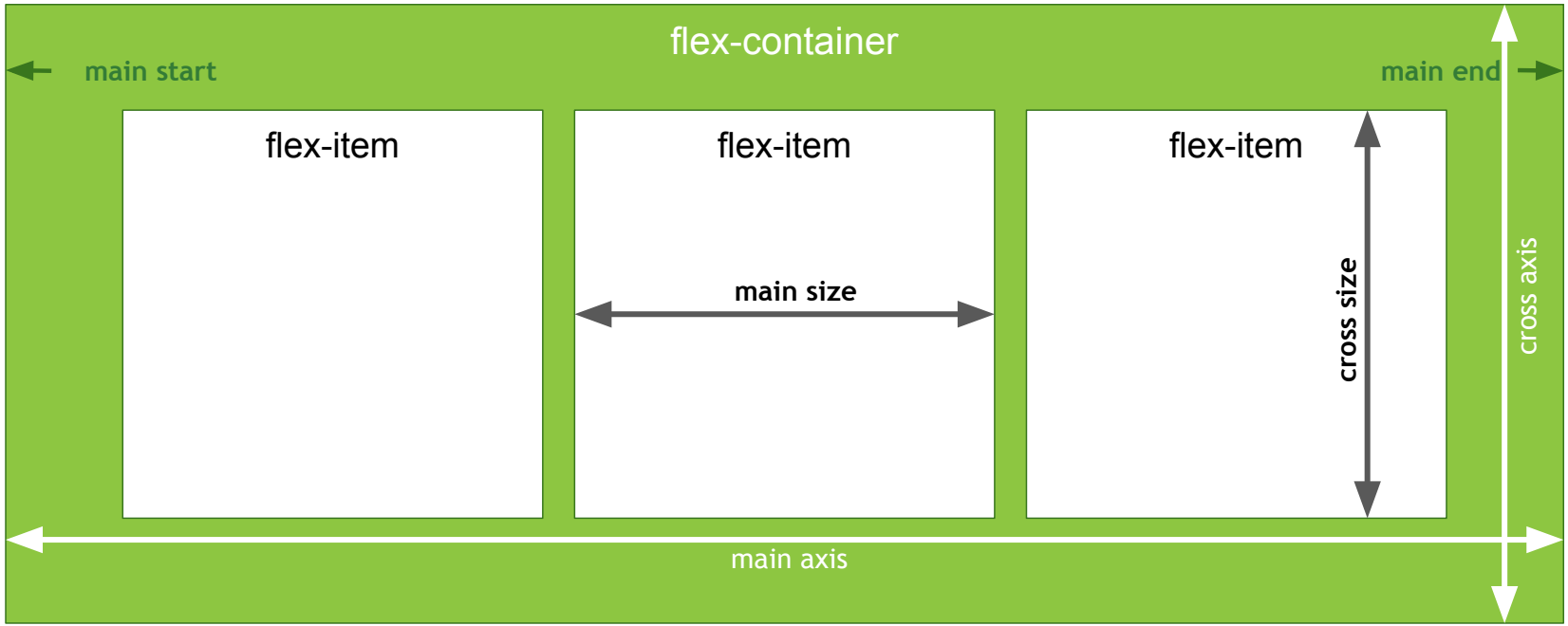
Estrutura do Flexbox



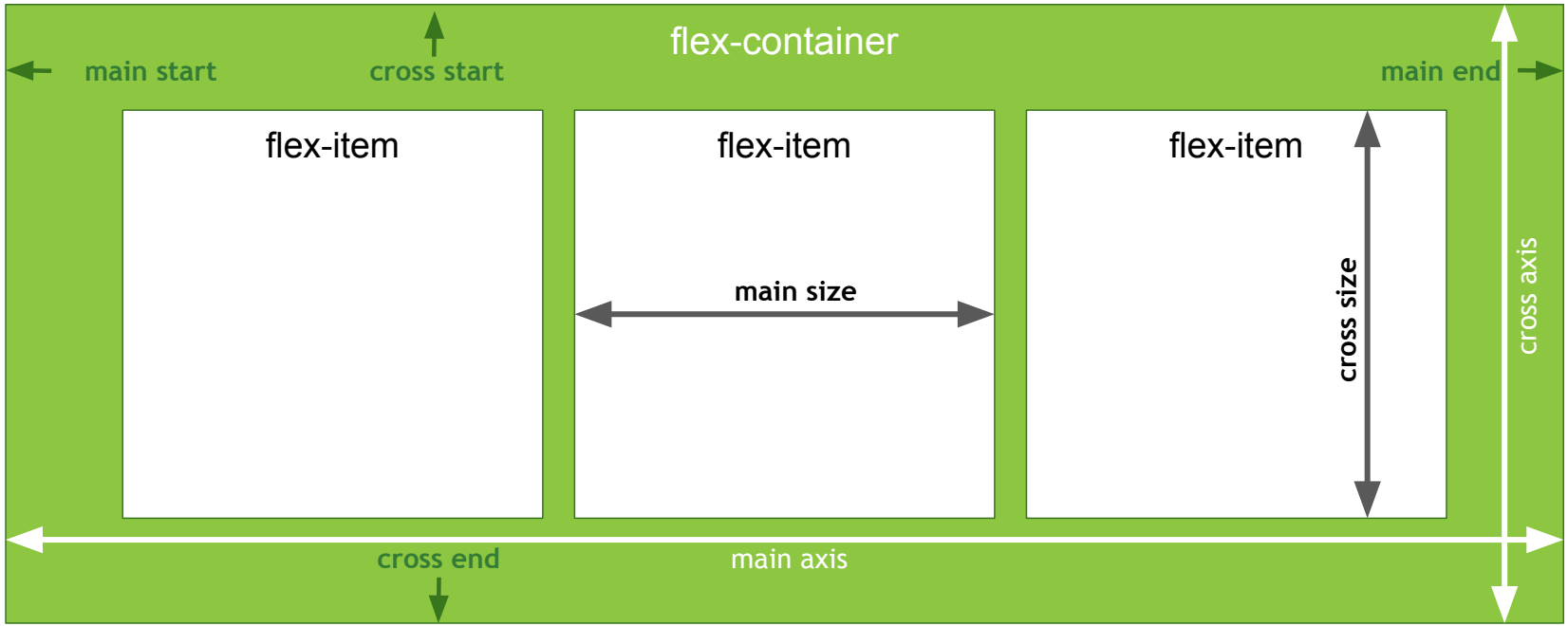
Estrutura do Flexbox



Estrutura do Flexbox



Estrutura do Flexbox



Propriedades do Flexbox

Coisas que podemos alterar por CSS

- flex container
  - flex-direction
  - flex-wrap
  - flex-flow
  - justify-content
  - align-items
  - align-content
  - …
- flex item
  - order ou flex-order
  - align-self
  - flex-grow
  - flex-shink
  - flex-basis
  - flex
  - …

Flex Box

```
.flex-container {
  display: flex;
}
```

- Como habilitar?
  - A partir do CSS3 o propriedade **display** permite mais uma opção: **flex**;
  - Atributo deve ser usado em um flex-container

Flex Box

- Criem um arquivo HTML e coloquem o código ao lado

```
<html>
    <head>
        <link rel="stylesheet"
href="estilo_flexbox.css">
    </head>
    <body>

        <div class="flex-container">
            <div>1</div>
            <div>2</div>
            <div>3</div>
            <div>4</div>
            <div>5</div>
            <div>6</div>
        </div>

    </body>
</html>
```

Flex Box

- Criem um arquivo CSS com o nome **estilo_flexbox.css** e coloquem o código ao lado

```
.flex-container {
   display: flex;
   background-color:  green;
}
.flex-container > div {
   background-color: #f1f1f1;
   margin: 10px;
   width: 100px;
   text-align: center;
   padding: 20px;
   font-size: 30px;
}
```

Flex Box

- Criem um arquivo CSS com o nome **estilo_flexbox.css** e coloquem o código ao lado
  - Abram o arquivo, o que vocês encontraram?

```
.flex-container {
   display: flex;
   background-color:  green;
}
.flex-container > div {
   background-color: #f1f1f1;
   margin: 10px;
   width: 100px;
   text-align: center;
   padding: 20px;
   font-size: 30px;
}
```

Flex Box

- Criem um arquivo CSS com o nome **estilo_flexbox.css** e coloquem o código ao lado
  - Abram o arquivo, o que vocês encontraram?

```
.flex-container {
   display: flex;
   background-color:  green;
}
.flex-container > div {
   background-color: #f1f1f1;
   margin: 10px;
   width: 100px;
   text-align: center;
   padding: 20px;
   font-size: 30px;
}
```

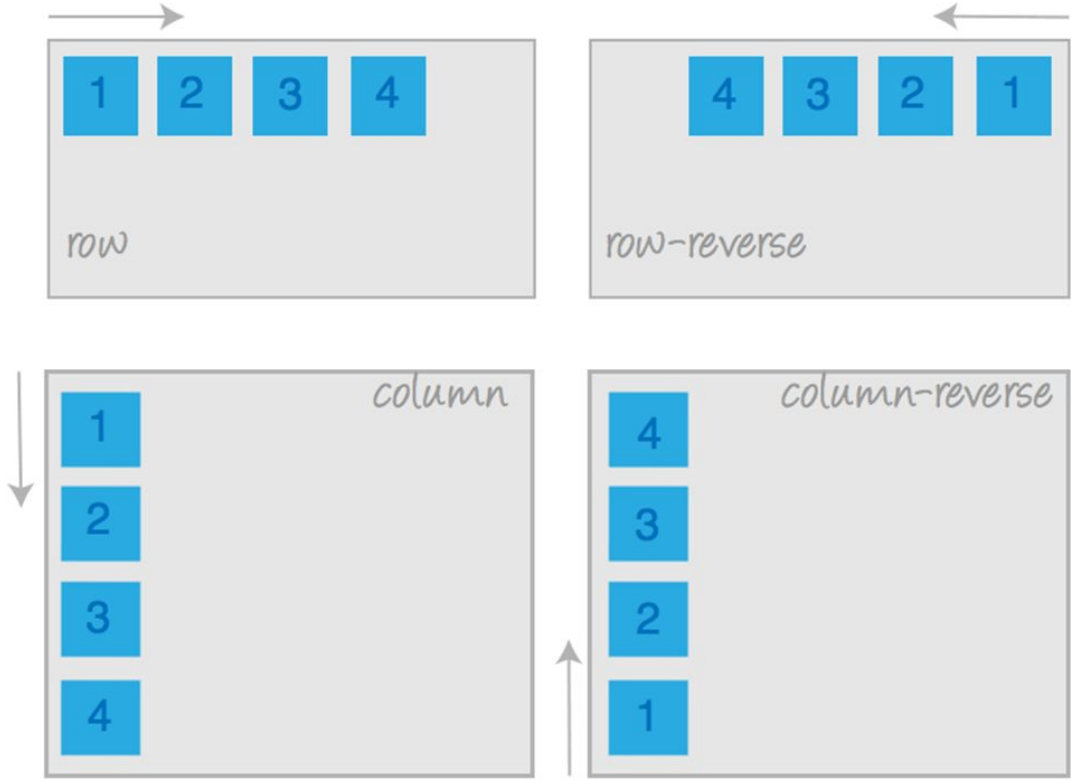
Propriedades do **Flex-Container**

Vamos usar o inspetor do navegador

Propriedade flex-direction

- flex-direction
  - Controla a direção dos itens dentro do container

```
flex-direction: row |
                row-reverse |
                column |
                column-reverse;
```



Flex Box

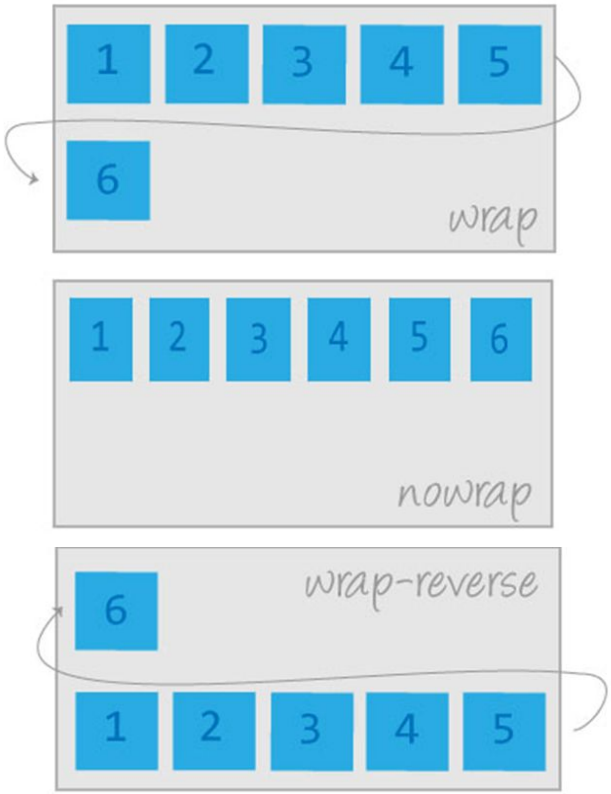
- Adicionem do flex-direction no CSS
  - Vejam os resultados!

```
.flex-container {
   display: flex;
   background-color:  green;
   flex-direction:  column;
}
```

Propriedade flex-wrap

- flex-wrap
  - controla se o flex-container é single-line ou multi-line e a direção do Cross Axis

```
flex-wrap: nowrap|
           wrap |
           wrap-reverse
```



Flex Box

```
.flex-container {
   display: flex;
   background-color:  green;
   flex-flow:  row wrap;
}
```

- **flex-flow**
  - permite declarar simultaneamente as propriedades **flex-direction** e **flex-wrap**

flex-flow: <'flex-direction'> <'flex-wrap'>;

Flex Box

```
.flex-container {
   display: flex;
   justify-content:  space-around;
}
```

- **justify-content**
  - controla o alinhamento dos **flex-items** em **relação ao main axis** – distribui o espaço não ocupado

```
justify-content: flex-start |
                 flex-end |
                 center |
                 space-between |
                 space-around;
```

Flex Box

```
.flex-container {
   display: flex;
   justify-content:  space-around;
}
```

- **justify-content**
  - controla o alinhamento dos **flex-items** em **relação ao main axis** – distribui o espaço não ocupado

```
justify-content: flex-start |
                 flex-end |
                 center |
                 space-between |
                 space-around;
```

Importante: é necessário que as medidas flexíveis e margens automáticas sejam removidas para que esta propriedade apresente o comportamento esperado

Flex Box

```
.flex-container {
   display: flex;
   justify-content:  space-around;
}
```

- **justify-content**
  - controla o alinhamento dos **flex-items** em **relação ao main axis** – distribui o espaço não ocupado

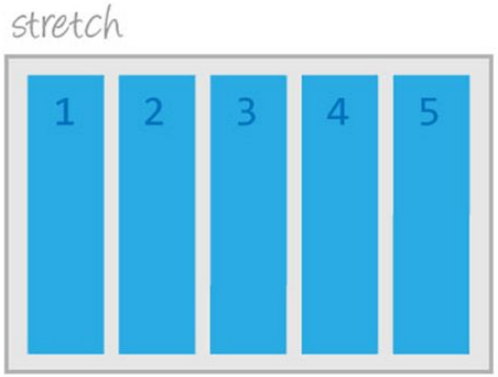

```
justify-content: flex-start |
                 flex-end |
                 center |
                 space-between |
                 space-around;
```

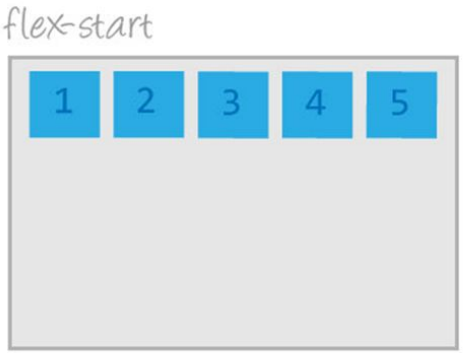
Propriedade align-items

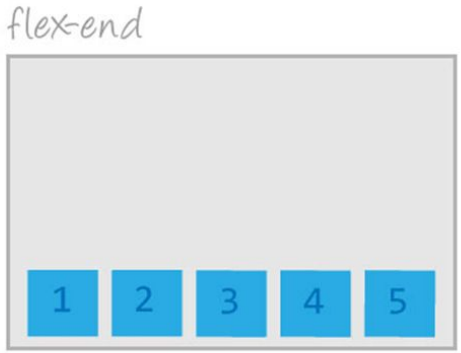
- **align-items**
  - ***similar*** a propriedade ***justify-content***, mas, ao invés de alinhar os itens através do main axis, o faz no ***cross axis***
  - Entre as opções aquela que possui um comportamento novo é a **baseline**
    - **baseline** alinha os flex-items tal como o alinhamento de sua linha de base (informalmente a linha base onde o texto se assenta)

```
align-items: flex-start |
             flex-end |
             center |
             baseline |
             stretch;
```

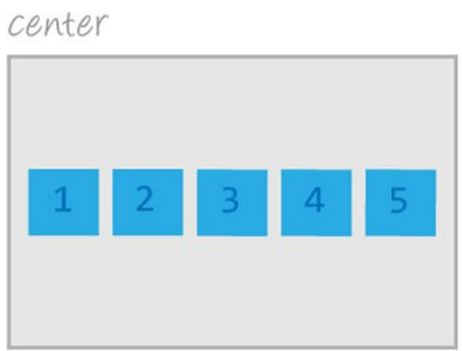
Propriedade align-items







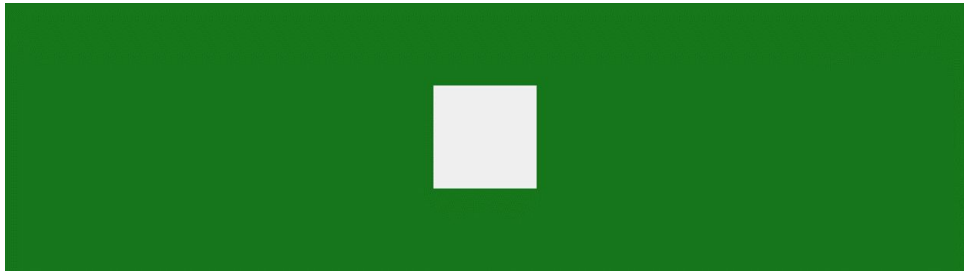

```
align-items: flex-start |
             flex-end |
             center |
             baseline |
             stretch;
```



Flex Box

```
.flex-container {
   display: flex;
   justify-content: center;
   align-items: center;
}
```

- As propriedades j**ustify-content e align-items** permitem exibir um centro perfeito na tela ao utilizar o valor **center**.



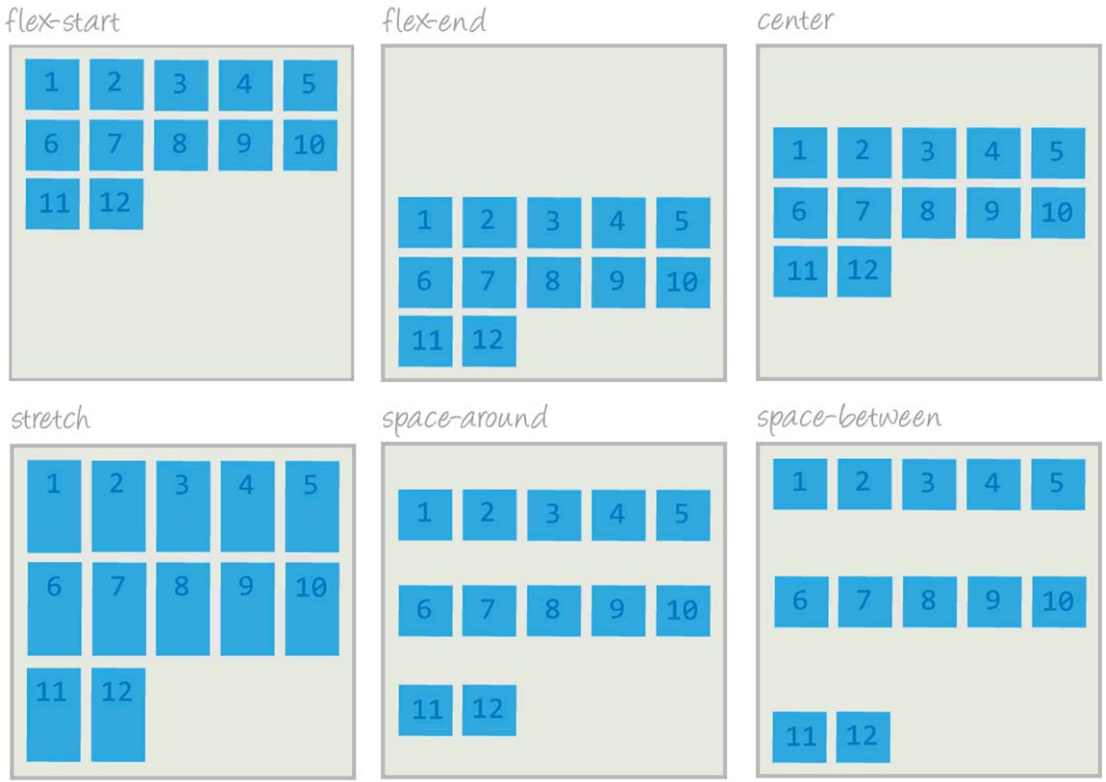
Propriedade align-content

- **align-content**
  - funciona de maneira ***semelhante ao justify-content***, contudo ***alinha as linhas*** ao invés dos flex-items individuais
  - Observação: essa propriedade não apresenta qualquer efeito quando o flex container tem somente 1 linha.

```
align-content: flex-start |
               flex-end |
               center|
               space-between |
               space-around |
               stretch;
```

Propriedade align-content

```
align-content: flex-start |
               flex-end |
               center|
               space-between |
               space-around |
               stretch;
```



Propriedades do **Flex-Items**

Vamos usar o inspetor do navegador

Propriedade order

- **Propriedade order**
  - Utiliza um ***valor inteiro*** para ***alterar a ordem*** dos ***flex-items***
  - Se a ordem tiver o mesmo valor, a ordem no código é utilizada
    - Por padrão a ordem em cada flex-item é 0
  - Aceita valores numéricos negativos

```
order: <integer>;
```

Flex Item

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```

- **Propriedade order**

Propriedade align-self

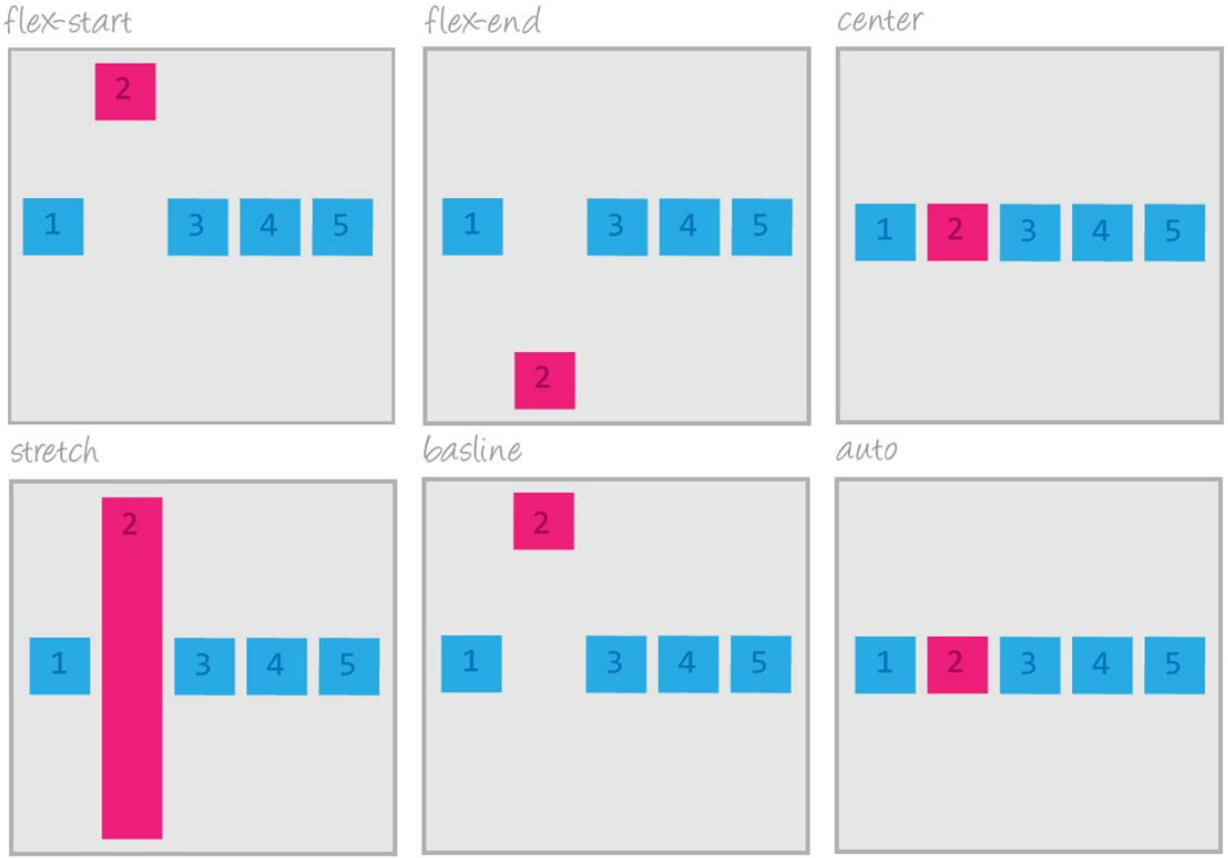
- **Propriedade align-self**
  - permite sobrescrever o comportamento do align-items em flex-items específicos alinhados no cross axis

```
align-self: auto |
            flex-start |
            flex-end |
            center |
            baseline |
            stretch;
```

```
<div class="flex-container">
  <div>1</div>
  <div style="align-self: center">2</div>
  <div class="itemEspecifico">3</div>
  <div>4</div>
</div>
```

```
.itemEspecifico {
   align-self: center;
}
```

Propriedade align-self: Comportamentos



Propriedade flex-grow

- **Propriedade flex-grow**
  - especifica o quanto um flex-item deve crescer em relação ao restante dos flex-items
  - o valor padrão é 0
  - valores negativos são inválidos

flex-grow: <number>;

Propriedade flex-grow


```
<div class="flex-container">
   <div style="flex-grow: 0">1</div>
   <div style="flex-grow: 1">2</div>
   <div style="flex-grow: 6">3</div>
</div>
```

```
.itemEspecifico {
   flex-grow: 1;
}
```

https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-grow

Propriedade flex-grow


```
<div class="flex-container">
   <div style="flex-shrink: 1">1</div>
   <div style="flex-shrink: 3">2</div>
   <div style="flex-shirnk: 0">3</div>
   <div >4</div>
   <div style="flex-shirnk: 3">5</div>
   <div >6</div>
   <div >10</div>
</div>
```

```
.itemEspecifico {
   flex-shrink: 1;
}
```

https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-shrink

Propriedade Flex

- **flex**
  - permite definir as propriedades f**lex-row, flex-shrink e flex-basis** na mesma propriedade de um **flex-item**
  - O valor inicial é **0 1 auto.**
  - As propriedades **flex-grow** e **flex-shrink** são opcionais e podem ser omitidas da declaração de flex.

flex: none | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]

```
.itemEspecifico {
   flex: 0 1 200px;
}
```

```
.itemEspecifico {
   flex: 200px;
}
```

Grid-view

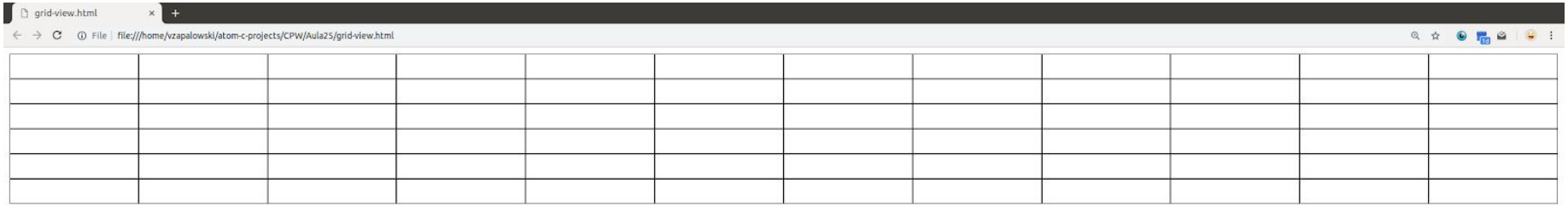
Grid-view

- O que é Grid-view?

Grid-view

- É um tipo de visualização
  - utilizada por muitos sites e frameworks
  - que é baseada na ideia de grids
  - grid ⇒ grade
    - então o site é dividido em uma grade de conteúdo onde podemos colocar as coisas

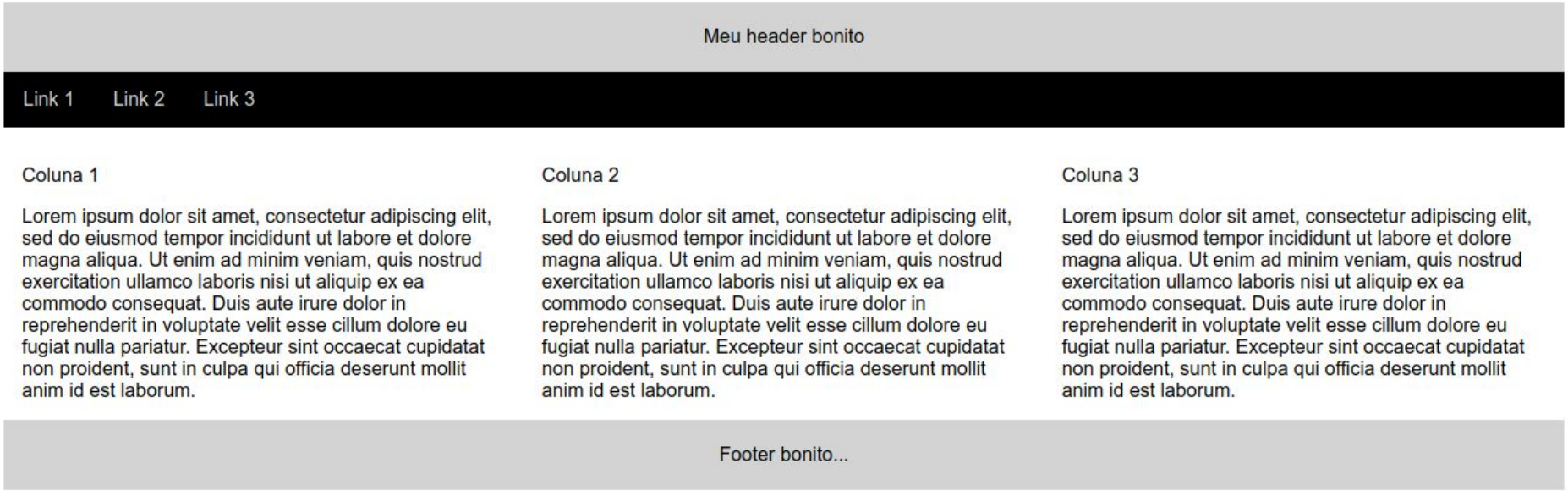
Como fazer algo assim?



Grid-view

- Geralmente o grid é dividido em 12 colunas
  - Cada coluna é um espaço a ser ocupado
  - Então para criar um grid-view como esse
    - O que a gente precisa fazer?

Na aula passada...



Na aula passada..

- Esse grid de 3 posições (colunas)
  - Lembra bastante o grid de 12 posições que precisamos fazer agora
  - **Como que a gente fez as colunas do layout da aula passada?**

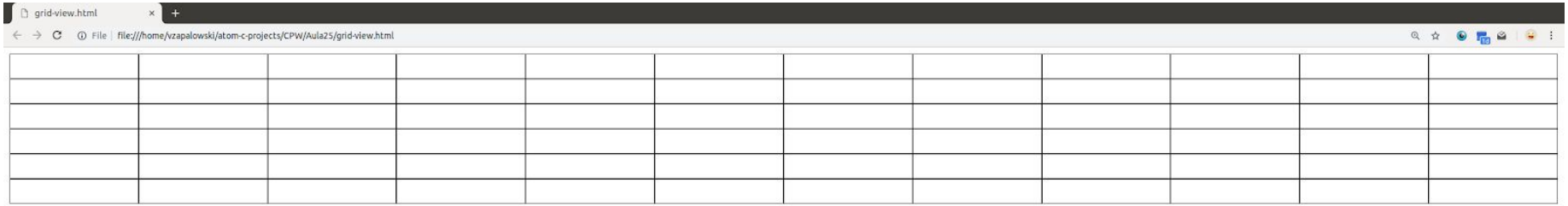
Na aula passada..

- Para criar colunas
  - largura de 31%
    - daí colocar no html 3 divs com colunas
  - ter o linha::after
    - para limpar a linha de floats

```
.coluna {
  float: left;
  width: 31%;
  padding: 15px;
}

.linha::after {
  content: "";
  display: block;
  clear: both;
}
```

Como fazer algo assim?



Como fazer o grid?

- O que a gente tem de fazer no HTML?
- O que a gente tem de fazer no CSS?

Como fazer o grid?

- O que a gente tem de fazer no HTML?

```
<div class="linha">
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
</div>
```

Como fazer o grid?

- Com base nesse HTML que a gente fez, o que a gente precisa fazer no CSS?

```
<div class="linha">
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
</div>
```

Coluna

- Com base nesse HTML que a gente fez, o que a gente precisa fazer no CSS?
  - **Como que a gente faz o estilo da classe col-1?**

- Com base nesse HTML que a gente fez, o que a gente precisa fazer no CSS?
  - **Como que a gente faz o estilo da classe col-1?**

```
.col-1 {
  float: left;
   border: 1px solid black;
   padding: 15px;
   width: 8.33%;
}
```

- Com base nesse HTML que a gente fez, o que a gente precisa fazer no CSS?
  - **Como que a gente faz o estilo da classe col-1?**
  - **Colunas de tamanho**
    - **1/12 = 0.083333**

```
.col-1 {
   float: left;
    border: 1px solid black;
    padding: 15px;
    width: 8.33%;
}
```

- Olhem a página de vocês como que tá…
- O que aconteceu aí?

```
.col-1 {
  float: left;
   border: 1px solid black;
   padding: 15px;
   width: 8.33%;
}
```

- Olhem a página de vocês como que tá…
- O que aconteceu aí?
  - **Aconteceu que alguma colunas pularam de linha, isso?**

```
.col-1 {
  float: left;
   border: 1px solid black;
   padding: 15px;
   width: 8.33%;
}
```

- Olhem a página de vocês como que tá…
- O que aconteceu aí?
  - **Aconteceu que alguma colunas pularam de linha, isso?**
    - **Por quê elas pularam de linha?**

```
.col-1 {
   float: left;
    border: 1px solid black;
    padding: 15px;
    width: 8.33%;
}
```

- É um atributo que define o tamanho da caixa do elemento html
- Existem dois valores
  - box-sizing : content-box
  - box-sizing : border-box

```
.col-1 {
   float: left;
    border: 1px solid black;
    padding: 15px;
    width: 8.33%;
}
```

- box-sizing : content-box
  - é o valor padrão
  - width e height são referentes ao conteúdo do elemento
    - não incluí padding, border

```
.col-1 {
  float: left;
   border: 1px solid black;
   padding: 15px;
   width: 8.33%;
}
```

- box-sizing : border-box
  - width e height incluem padding, border e conteúdo de fato
  - Por isso que dava problema na classe col-1
  - width era mais que 8.33%

```
.col-1 {
   float: left;
    border: 1px solid black;
    padding: 15px;
    width: 8.33%;
}
```

- Para fazer com que o grid fique exatamente com 100% de acordo com a largura da página
- Inserir o estilo do box-sizing
- Deve fazer com que ocupe a linha toda...

```
.col-1 {
   float: left;
    border: 1px solid black;
   padding: 15px;
    width: 8.33%;
    box-sizing: border-box;
}
```

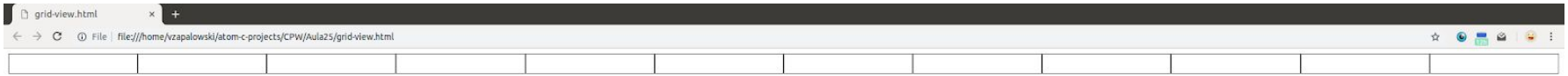
- Com base nesse HTML que a gente fez, o que a gente precisa fazer no CSS?
  - Como que a gente faz o estilo da classe col-1?
  - **Como que a gente faz o estilo da classe linha?**

```
<div class="linha">
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
</div>
```

- Vimos que geralmente existe a classe o clearfix
- No layout da aula tinha a mesma classe linha que a gente precisa aqui

```
.col-1 {
   float: left;
    border: 1px solid black;
    padding: 15px;
    width: 8.33%;
}
.linha::after {
   content: "";
   display: block;
   clear: both;
}
```

Até agora...


grid-view.html
File | file:///home/vzapalowski/atom-c-projects/CPW/Aula25/grid-view.html

- Colocar um parágrafo em algumas das colunas
- Para ver o que acontece com o conteúdo da página
- **O que acontece?**

```
<div class="linha">
  <div class="col-1"></div>
  <div class="col-1">
<p>Teste da coluna 2 que fica com o texto grande e bonito...</p></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1">
<p>Teste da coluna 6 que fica feia, boba e chata...</p></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
```

Até agora...


grid-view.html
File | file:///home/vzapalowski/atom-c-projects/CPW/Aula25/grid-view.html
Teste da coluna 2 que fica com o texto grande e bonito...
Teste da coluna 6 fica feia, boba e chata...

Mais espaço na coluna...

- Se eu quiser dividir 1 coluna das 12
  - Ok, é só colocar a classe col-1
  - **Mas e se eu quiser usar o espaço de mais de uma coluna?**
    - Como se a gente fosse mesclar uma coluna…
    - **Como que a gente faria?**

Até agora...


grid-view.html
File | file:///home/vzapalowski/atom-c-projects/CPW/Aula25/grid-view.html
Teste da coluna 2 que fica com o texto grande e bonito...
Teste da coluna 6 fica feia, boba e chata...

Mais espaço na coluna...

- **Como que vocês acham que a gente faria?**
  - **Aceito sugestões...**

- Para fazer o grid o importante era ter 12 divs com a classe col-1…
- Isso obrigatoriamente vai ocupar 100% da largura da tela
  - 12 * 8.33% = 99.96%

```
<div class="linha">
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
</div>
```

- Para ter uma coluna que tenha a largura de duas colunas...o que a gente precisa fazer?

```
<div class="linha">
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
  <div class="col-1"></div>
</div>
```

- Para ter uma coluna que tenha a largura de duas colunas...o que a gente precisa fazer?
  - **Criar uma classe de duas colunas!**

```
<div class="linha">
   <div class="col-1"></div>
   <div class="col-1"><p>Teste da
coluna 2 que fica com o texto grande
e bonito…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-2"><p>Teste da
coluna 6 que fica feia, boba e
chata…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
</div>
```

- Para ter uma coluna que tenha a largura de duas colunas...o que a gente precisa fazer?
  - **Criar uma classe de duas colunas!**
  - **Deletar uma coluna col-1**

```
<div class="linha">
   <div class="col-1"></div>
   <div class="col-1"><p>Teste da
coluna 2 que fica com o texto grande
e bonito…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-2"><p>Teste da
coluna 6 que fica feia, boba e
chata…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
</div>
```

- Importante é continuar com 12 colunas
  - Somando todas as divs
  - 10 divs de col-1
  - 1 div de col-2
- **Como é o CSS da classe col-2?**

```
<div class="linha">
   <div class="col-1"></div>
   <div class="col-1"><p>Teste da
coluna 2 que fica com o texto grande
e bonito…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-2"><p>Teste da
coluna 6 que fica feia, boba e
chata…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
</div>
```

- O estilo vai ser praticamente igual ao da col-1
- Só que a gente vai alterar a largura dela para 16.66%
  - Assim ela vai ocupar o espaço de 2 colunas…
  - Testem aí...

```
.col-1 {
    float: left;
     border: 1px solid black;
     padding: 15px;
     width: 8.33%;
     box-sizing: border-box;
}
.col-2 {
    float: left;
     border: 1px solid black;
     padding: 15px;
     width: 16.66%;
     box-sizing: border-box;
}
.linha::after {
  content: "";
  display: block;
  clear: both;
}
```

- Para ter uma coluna que tenha a largura de três colunas...o que a gente precisa fazer?
  - **Criar uma classe de três colunas!**

```
<div class="linha">
   <div class="col-1"></div>
   <div class="col-1"><p>Teste da
coluna 2 que fica com o texto grande
e bonito…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-3"><p>Teste da
coluna 6 que fica feia, boba e
chata…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
</div>
```

- Para ter uma coluna que tenha a largura de três colunas...o que a gente precisa fazer?
  - **Criar uma classe de três colunas!**
  - **Deletar mais uma coluna col-1**

```
<div class="linha">
   <div class="col-1"></div>
   <div class="col-1"><p>Teste da
coluna 2 que fica com o texto grande
e bonito…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-3"><p>Teste da
coluna 6 que fica feia, boba e
chata…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
</div>
```

- Importante é continuar com 12 colunas
  - Somando todas as divs
  - 9 divs de col-1
  - 1 div de col-3
- Como é o CSS da classe col-3?

```
<div class="linha">
   <div class="col-1"></div>
   <div class="col-1"><p>Teste da
coluna 2 que fica com o texto grande
e bonito…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-3"><p>Teste da
coluna 6 que fica feia, boba e
chata…</p></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
   <div class="col-1"></div>
</div>
```

- O estilo vai ser praticamente igual ao da col-1
- Só que a gente vai alterar a largura dela para 25%
  - Assim ela vai ocupar o espaço de 3 colunas…
  - Testem aí...

```
.col-1 {
   float: left;
    border: 1px solid
black;
    padding: 15px;
    width: 8.33%;
}
.col-2 {
   float: left;
    border: 1px solid
black;
    padding: 15px;
    width: 16.66%;
}
```

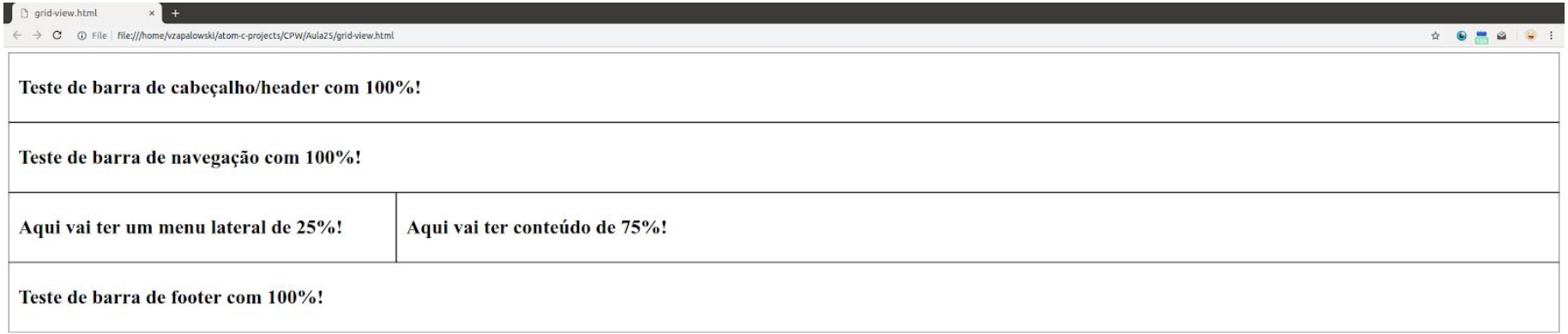
```
.col-3 {
   float: left;
    border: 1px solid
black;
    padding: 15px;
    width: 25%;
}
.linha::after {
  content: "";
  display: block;
  clear: both;
}
```

- Para criarmos o grid completo
- Vamos criar do col1 até o col12
- Onde a classe
  - col1 ocupa 1/12 de largura
  - col12 ocupa 100% de largura

```
.col-1 { ...
   width: 8.33%;
}
.col-2 { ...
   width: 16.66%;
}
.col-3 { ...
   width: 25%;
}
.col-4 { ...
   width: 33.33%;
}
.col-5 { ...
   width: 41.66%;
}
.col-6 { ...
   width: 50%;
}
```

```
.col-7 { ...
   width: 58.33%;
}
.col-8 { ...
   width: 66.66%;
}
.col-9 { ...
   width: 75%;
}
.col-10 { ...
   width: 83.33%;
}
.col-11 { ...
   width: 91.66%;
}
.col-12 { ...
   width: 100%;
}
```

Qual HTML para gerar essa página?



Pseudo-element

Pseudo-element

- O que é pseudo-element?

Pseudo-element

- Seletor
  - É mais uma forma de selecionar conteúdo do html pelo CSS
  - Só que mais específico ainda do que os seletores anteriores
    - Podemos selecionar partes de um elemento
    - Podemos inserir conteúdo no html

Pseudo-element

- Sintaxe é
  - **seletor::pseudo-element { … }**
    - ou seja, é dois pontos e o pseudo-seletor que queremos

Pseudo-element

- Coloque uma parágrafo um pouco maior no html
- E o estilo ao lado
- **O que acontece aí na página?**

```
p::first-line {
  color: red;
  text-transform:
uppercase;
}
```

Pseudo-element

- Então…
  - Pega a primeira linha do parágrafo e aplica o estilo
  - **Redimensionem o tamanho da página para ver o que acontece**

```
p::first-line {
  color: red;
  text-transform:
uppercase;
}
```

Pseudo-element

- Troquem o estilo agora
- **O que acontece?**

```
p::first-letter {
  color: red;
  font-weight: bold;
  text-transform:
  uppercase;
}
```

Pseudo-element

- Troca somente a primeira letra do parágrafo
- **Uma das funcionalidades dos pseudo-elementos é trocar partes do elemento**

```
p::first-letter {
   color: red;
   font-weight: bold;
   text-transform:
   uppercase;
}
```

Pseudo-element

- Além de alterar os elementos
- **Coloquem o estilo ao lado**
- **O que acontece?**

```
p {
   font-size: 20;
}
p::before {
   content:
url("../imgs/bg1.jpg");
   display: block;
}
```

Pseudo-element

- O CSS insere uma imagem antes do parágrafo…
  - **::before**
- O CSS insere uma imagem depois do parágrafo…
  - **::after**

```
p {
    font-size: 20;
}
p::before {
  content: url("../imgs/bg1.jpg");
  display: block;
}
p::after {
    content: "é verdade esse bilhete...";
    font-weight: bold;
}
```

Pseudo-element

```
::selection {
  background: pink;
  color: blue;
}
```

- Além dessas coisas…
- Coloquem o estilo ao lado
- **O que acontece?**

Pseudo-element

- Ao selecionar algo
  - um texto, por exemplo
  - o estilo é alterado para o que está no ::selection

```
::selection {
  background: pink;
  color: blue;
}
```

Pseudo-element

- Existem mais alguns
  - ::placeholder
  - ::backdrop
  - …
- Só que **são experimentais** e sem garantia de suporte nos navegadores

Float

Float

- O que é float?

Float

- É como um elemento pode se posicionar/flutuar dentro do html em relação aos outros elementos
  - **float: left**
  - **float: right**
  - **float: none**

Float

```
img {
   width: 33%;
}
```

- Limpem o arquivo CSS
- Limpem o body do arquivo HTML também
- Coloquem no html
  - um parágrafo
  - uma imagem
- Coloque o estilo ao lado no css

Float

```
img {
   width: 33%;
}
```

- O que acontece na tela?

Float

- **é o valor padrão de float**
  - como ***none*** o elemento não sai do lugar ocupando a posição estática dele
  - como todos elementos que vimos até agora

```
img {
   width: 33%;
    float: none;
}
```

Float

- O que acontece na tela?

```
img {
   width: 33%;
    float: left;
}
```

Float

- A imagem passa a flutuar pela tela de acordo com o valor do float que colocamos
- Alterem para right para ver o que acontece...

```
img {
   width: 33%;
    float: right;
}
```

Float

- Então, o float é importante para que a gente consiga colocar elementos melhor posicionados na tela
- Isso vale para todos os elementos
  - **Se trocar o img por p?**
    - **E colocar mais um p no html**

```
p {
   width: 33%;
    float: right;
}
```

Float

- Voltando para o exemplo da imagem
  - colocar no html

```
<div>
  <img src="imgs/bg1.jpg"/>
  <p> Teste de float! </p>
</div>
<p id="zoado"> Teste de float zoado! </p>
```

Float

- Colocar no CSS
- **O que acontece com a página?**

```
p {
  width: 33%;
  float: left;
}
img {
  width: 33%;
  float: left;
}
div {
  border: solid 2px black;
}
```

Clear

Clear

- O que é clear?

Clear

- é o que é permitido fazer float sobre o elemento que está sendo aplicado o clear
  - clear: none
  - clear : left;
  - clear: right;
  - clear: both;

- clear:none;
- é o padrão
- quer dizer que podem aparecer elementos float tanto na esquerda quanto na direita
- **por isso que o parágrafo ficava no lugar errado na outra página**

```
p {
  width: 33%;
  float: left;
}
img {
  width: 33%;
  float: left;
}
div {
  border: solid 2px black;
}
```

- **Como que a gente pode resolver isso?**

Clear

- Temos de bloquear os floats para que os elementos html parem de se colocar um do lado do outro…
- clear: both;
  - bloqueia elemento de aparecerem com float: right e left

```
#zoado {
   clear:both;
}
```

Clear

- clear: both
  - bloqueia ambos os lados de float
  - **o que aconteceu aí na página?**

```
#zoado {
   clear:both;
}
```

Clear

```
#zoado {
   clear:both;
}
```

- Isso causa dois problemas
  - **Alguma ideia de quais?**

Clear

- Primeiro é ter de ficar lembrando de colocar clear:both sempre quando fechar a linha do html
- Segundo é que o espaço do div também ficou estranho. Da para ver pela borda dele

```
#zoado {
   clear:both;
}
```

Clear

- Para resolver esse problema do html se perder com float
  - uma classe de estilo que é muito comum
  - **clearfix junto do after**
    - **adicionem no CSS**
    - **o que isso faz?**

```
.clearfix::after {
  content: "";
  clear: both;
  display: block;
}
```

Clear

- Tirem o #zoado do CSS
- Em qual elemento eu coloco a classe clearfix?

```
<div>
  <img src="imgs/bg1.jpg"/>
  <p> Teste de float! </p>
</div>
<p id="zoado"> Teste de float
zoado! </p>
```

Clear

- **O que acontece na tela?**

```
<div class="clearfix">
  <img src="imgs/bg1.jpg"/>
  <p> Teste de float! </p>
</div>
<p id="zoado"> Teste de float
zoado! </p>
```

Clear

- **O que acontece na tela?**
  - Agora fica tudo certinho
  - Porque o clearfix coloca sempre um elemento depois do que tem a classe (::after)

```
<div class="clearfix">
  <img src="imgs/bg1.jpg"/>
  <p> Teste de float! </p>
</div>
<p id="zoado"> Teste de float
zoado! </p>
```

MUITO

OBRIGADO