Layouts com CSS

Disciplina: Desenvolvimento Web

Prof. Dr. Rafael Will M. de Araujo

Sumário

- Fazer layouts com o CSS
- Dimensões no CSS
- Como funciona o *display*
- Como funciona uma *media query*
- Dois métodos de layout modernos: *Flexbox* e *Gridlayout*

Conteúdo

Dimensões

- Estas propriedades definem o tamanho dos elementos renderizados. Elas são height e width;
- Recebe valores em tamanho e porcentagem (mesmas unidades vistas anteriormente).

Exemplo

```
1 seletor {
2     height: valor;
3     width: valor;
4 }
```

Dimensões

- Exemplo:

CSS

```css
div {
  border: 1px solid black;
}

div.div1 {
  height: 200px;
  width: 25%;
  background: yellow;
}

div.div2 {
  height: 100px;
  width: 5em;
  background: red;
}
```

HTML

```html
<div class="div1">div 1</div>
<div class="div2">div 2</div>
```

Exemplo: https://codepen.io/rafaelwill/pen/JjJEeXO

Classificação

- Estas propriedades são usadas para classificar os elementos e definir como eles devem aparecer no layout;
  - *float*
  - *clear*
  - *display*

Float

- Define onde um elemento aparecerá em outro elemento;
- Pode assumir os valores *left* ou *right*;
- O elemento com float irá mover-se mais a esquerda ou mais a direita possível.

Exemplo

```
1 seletor {
2     float: valor;
3 }
```

Float

- Exemplo:

CSS

```css
.div1 {
  float: left;
  background-color: gray;
}

.div2 {
  float: left;
  background-color: cyan;
}

.div3 {
  float: right;
  background-color: gray;
}

.div4 {
  float: right;
  background-color: cyan;
}
```

HTML

```html
<div class="div1">div 1</div>
<div class="div2">div 2</div>
<div class="div3">div 3</div>
<div class="div4">div 4</div>
```

Exemplo: https://codepen.io/rafaelwill/pen/oNwBQeM

Clear

- Define em quais lados de um elemento outros elementos não podem flutuar (*float*);
- Assume valores: *none*, *left*, *right*, *both*.

Exemplo

```
1 seletor {
2     clear: valor;
3 }
```

Clear

- Exemplo:

CSS

```css
.div1 {
  float: left;
  background-color: gray;
}

.div2 {
  float: left;
  clear: left;
  background-color: cyan;
}
```

HTML

```html
<div class="div1">div 1</div>
<div class="div2">div 2</div>
```

Exemplo: https://codepen.io/rafaelwill/pen/QWgdJqy

Display

- Define a maneira como os diversos elementos de uma página são exibidos. Ele especifica o comportamento que o elemento terá no momento que for "renderizado".

Exemplo

```
1 seletor{
2     display: valor;
3 }
```

- Temos quatro valores básicos:
  - **block**: define uma região em bloco, que possui altura definida pelo conteúdo (ou CSS) e se expande da esquerda para a direita até onde for possível.
    - O elemento começará em uma nova linha e ocupará toda a largura disponível. Permite definir os valores de largura e altura.
  - **inline**: define uma região em linha, com a largura definida pelo conteúdo e altura pelo tamanho da linha.
    - O elemento não começa em uma nova linha e ocupa apenas a largura necessária. Não é possível definir a largura ou altura.
  - **inline-block**: define uma região com tamanho controlada pelo conteúdo ou CSS. Nenhuma *tag* funciona assim por padrão.
    - É formatado como um elemento `inline`, isto é, ele não começa em uma nova linha. Mas, permite definir valores de largura e altura.
  - **none**: define uma região invisível, ela existe no HTML mas não aparece para o usuário e não altera o layout.

Display

- Exemplo: https://codepen.io/rafaelwill/pen/WNORYXQ

CSS

```css
.invisivel {
  display: none;
}

.block {
  display: block;
  background-color: gray;
}

.inline {
  display: inline;
  background-color: cyan;
}
```

HTML

```html
<div class="invisivel">div invisivel</div>
<div class="block">div block</div>
<div class="inline">div inline</div>
<div class="inline">outra div inline</div>
```

Display

- Exemplo 2: https://codepen.io/rafaelwill/pen/dyRNQZg

CSS

```css
div {
  border: 3px solid black;
  width: 100px;
  height: 100px;
}

div.div1 {
  background: yellow;
  display: block;
}

div.div2 {
  background: red;
  display: inline;
}

div.div3 {
  background: green;
  display: inline-block;
}

div.div4 {
  background: black;
  display: none;
}
```

HTML

```html
<div class="div1">div 1</div>
<div class="div2">div 2</div>
<div class="div3">div 3</div>
<div class="div4">div 4</div>
```

Conteúdo

Box-Model

- Além da altura (*height*) e largura (*width*), temos algumas outras medidas que influenciam no tamanho total que um elemento HTML ocupa na página:
  - **Border**: já mostramos a borda, ela possui uma espessura, o que altera também o espaço ocupado pela *tag* HTML;
  - **Padding**: espaço entre o conteúdo da *tag* e a borda dela. Pode ser colocado para as quatro direções ou apenas as que forem interessantes (*top*, *bottom*, *left*, *right*);
  - **Margin**: espaço entre a borda do elemento até a borda do próximo elemento (ou de algum limite do documento). Também pode ser utilizada em todas as quatro direções ou apenas nas que forem interessantes.
- Exemplo:

Box-Model: border

- Exemplo: https://codepen.io/rafaelwill/pen/jOwyXrL

CSS

```css
div {
  border: 3px solid black;
  width: 100px;
  height: 100px;
}

div.div1 {
  background: yellow;
}

div.div2 {
  background: red;
  border-width: 10px;
}
```

HTML

```html
<div class="div1">div 1</div>
<div class="div2">div 2</div>
```

Box-Model: padding

- Exemplo: https://codepen.io/rafaelwill/pen/oNwBJLr

CSS

```css
div {
  border: 3px solid black;
  width: 100px;
  height: 100px;
}

div.div1 {
  background: yellow;
  padding: 10px;
}

div.div2 {
  background: red;
  padding: 20px;
}
```

HTML

```html
<div class="div1">div 1</div>
<div class="div2">div 2</div>
```

Box-Model: margin

- Exemplo: https://codepen.io/rafaelwill/pen/abwpPBP

CSS

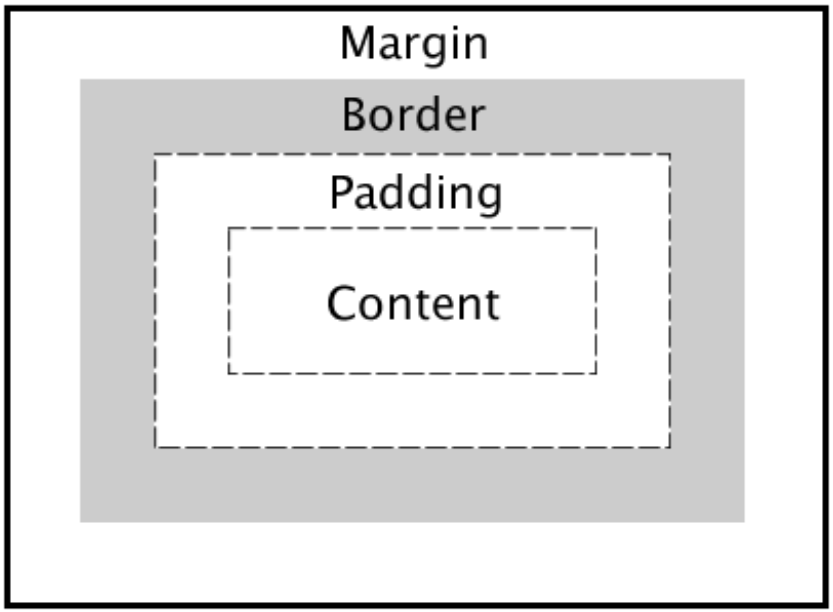
```css
div {
  border: 3px solid black;
  width: 100px;
  height: 100px;
}

div.div1 {
  background: yellow;
  margin: 10px
}

div.div2 {
  background: red;
  margin: 20px;
}
```

HTML

```html
<div class="div1">div 1</div>
<div class="div2">div 2</div>
```

Box-Model

- Essas distâncias configuram o que chamamos de Box-Model (modelo caixa):



- Desta maneira, quando definimos os atributos *height* e *width* do elemento em si, esses valores não levam em consideração o *padding* e o *border*, fazendo com que os elementos sejam maiores do que deveriam.

Box-Model

- Exemplo: https://codepen.io/rafaelwill/pen/yLXgGMK

CSS

```css
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
}

.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid blue;
}
```

HTML

```html
<div class="div1">div menor (width: 300px e height: 100px)</div>
<div class="div2">div maior (width: 300px e height: 100px)</div>
```

- Note que a div2 será maior que a div1. Isso ocorre porque, por padrão, o CSS não considera *padding*, *border* e *margin* no cálculo final do tamanho do elemento.

Box-Model

- Para corrigir o problema basta utilizar a propriedade *box-sizing*. Os valores possíveis são:
  - *content-box*: esse é o estilo padrão, conforme especificado pela norma CSS. As propriedades largura e altura (*width* e *height*) são medidas incluindo só o conteúdo do elemento, sem considerar *padding, border, margin*.
  - *border-box*: Indica que *padding* e *border* serão incluídos no cálculo das medidas de altura e largura. Detalhe: a propriedade *margin* não é considerada neste cálculo!

CSS

```css
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
}

.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  box-sizing: border-box;
  border: 1px solid blue;
}
```

HTML

```html
<div class="div1">div menor (width: 300px e height: 100px)</div>
<div class="div2">div maior (width: 300px e height: 100px)</div>
```

Exemplo: https://codepen.io/rafaelwill/pen/vYZgvmO

Box-Model

- Uma atitude comum entre os desenvolvedores é utilizar o *border-box* para todos os objetos como padrão. Para facilitar, podemos utilizar a seguinte regra para **colocar esse atributo para todos os elementos da página**:

Exemplo

```
1 * {
2     box-sizing: border-box;
3 }
```

Conteúdo

Media Query

- O conceito principal de uma *Media Query* é verificar algumas informações do dispositivo visualizando o site, assim podendo definir um conjunto de regras que funcione melhor para este dispositivo;
- É possível verificar principalmente:
  - Largura e altura do *viewport* (área visível da página);
  - Largura e altura do aparelho em si (tela completa);
  - Orientação (retrato ou paisagem);
  - Resolução
- Técnica de *Media Query* é muito popular no desenvolvimento mobile;
- Sintaxe básica:

Exemplo

```
1 @media not|only tipomedia and (mediafeature) {
2   Regras CSS
3 }
```

- Mais informações: https://developer.mozilla.org/pt-BR/docs/Web/CSS/Media_Queries/Using_media_queries

Media Query

- Neste exemplo, as áreas das classes div1 e div2 possuem um tamanho especificado.
- Quando o aparelho possuir resolução de tela (*screen*) com largura máxima de 700px, os tamanhos passam a ser definidos por 100% (tamanho total da janela).
- O marcador *only* diz que essa regra se aplica **apenas** ao *mediatype* (*screen*) e *mediafeature* (*max-width: 700px*).

CSS

```
1  .div1 {
2    width: 200px;
3    background: cyan;
4  }

5  .div2 {
6    width: 1000px;
7    background: yellow;
8  }

9  @media only screen and (max-width: 700px) {
10   .div1 {
11     width: 100%;
12   }

13   .div2 {
14     width: 100%;
15   }
16 }
```

HTML

```
1  <div class="div1">div 1</div>
2  <div class="div2">div 2</div>
```

Media Query

- Exemplo 2: https://codepen.io/rafaelwill/pen/gORgZjg

CSS

```css
#main {margin-left: 4px;}

#leftsidebar {
  float: none;
  width: auto;
}

#menulist {
  margin: 0;
  padding: 0;
}

.menuitem {
  background: #CDF0F6;
  border: 1px solid #D4D4D4;
  border-radius: 4px;
  list-style-type: none;
  margin: 4px;
  padding: 2px
}

@media screen and (min-width: 480px) {
  #leftsidebar {
    float: left;
    width: 200px;
  }

  #main {margin-left: 216px;}
}
```

Media Query

- continuação do Exemplo 2:

HTML

```
1  <div>
2    <div id="leftsidebar">
3      <ul id="menulist">
4        <li class="menuitem">Item 1</li>
5        <li class="menuitem">Item 2</li>
6        <li class="menuitem">Item 3</li>
7        <li class="menuitem">Item 4</li>
8        <li class="menuitem">Item 5</li>
9      </ul>
10   </div>
11   <div id="main">
12     <h1>Redimensione o navegador e veja o que acontece!</h1>
13     <p>Este exemplo mostra um menu a esquerda do conteúdo principal quando
14     o site tem pelo menos 480px de largura. Experimente ver o que acontece quando a tela fica menor.</p>
15   </div>
16 </div>
```

Conteúdo

Flexbox

- No CSS3 foi definida mais uma propriedade *display*, a **Flex(box)**. A ideia do Flexbox é garantir que o conteúdo interno da página consiga se reorganizar de uma maneira previsível caso a janela mude de tamanho;
- Ajuda a fazer layouts de maneira mais simplificada e fluída;
- Propriedades que fazem parte do modelo flexbox são divididas em propriedades para o container (que possui o display **flex**, chamado de **flex-container**) e para os seus filhos (chamados de **flex-items**);

Flexbox

- Para auxiliar em deixar previsível o comportamento dos elementos de um *flexbox-container* (em vermelho), temos uma série de propriedades envolvidas:
  - **flex-direction**: direção que os itens devem seguir (coluna-linha);
  - **justify-content**: como os itens se dispersam horizontalmente*;
  - **align-items**: como os itens se dispersam verticalmente*;
  - **flex-wrap**: se os itens devem passar a próxima linha caso falte espaço;
  - **align-content**: como as linhas se dispersam;
- Para os *flex-items* (em amarelo), precisamos das seguintes propriedades:
  - **order**: identifica a ordem que o item deve aparecer;
  - **align-self**: sobrecarrega o conteúdo da propriedade align-items;
  - **flex**: identifica o tamanho deste elemento, em relação aos demais.

- Para aprender um pouco sobre o Flexbox: http://flexboxfroggy.com

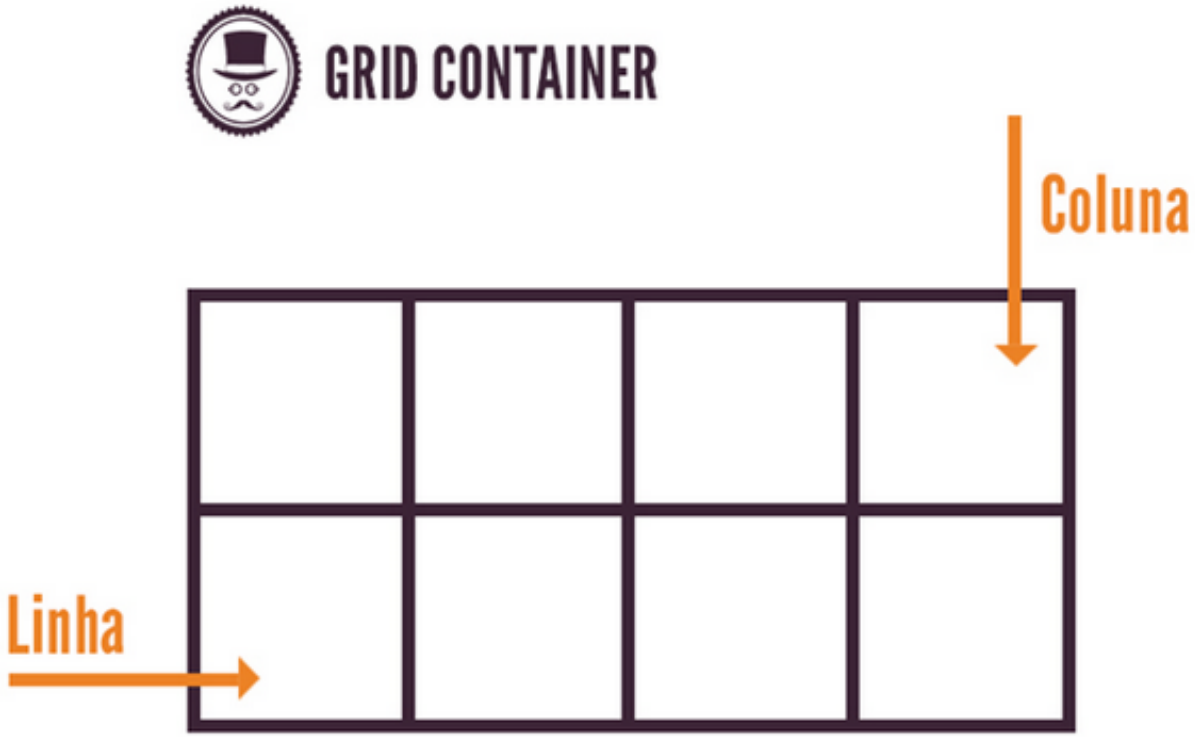
Grid Layout

- Apesar do Flexbox já ter adicionado uma ótima possibilidade de layouts fluídos, ele funciona apenas para containers em uma dimensão (linha ou coluna);
- Para obter layouts mais complicados, é necessário combinar vários containers com flexbox;
- Seria interessante possuir um sistema que funcionasse como uma tabela do HTML (definição de regiões), mas que não dependesse de tabelas do HTML;
- A partir do CSS3 criou-se o Grid-Layout, uma forma de construir o layout como se fosse uma tabela, ou um grid, sem usar as tabelas ou outros "hacks"de CSS;
- Vale lembrar que a sua utilização é um tanto restrita: nem todos os navegadores implantaram toda sua especificação (leia-se: Opera e Edge).

Grid Layout

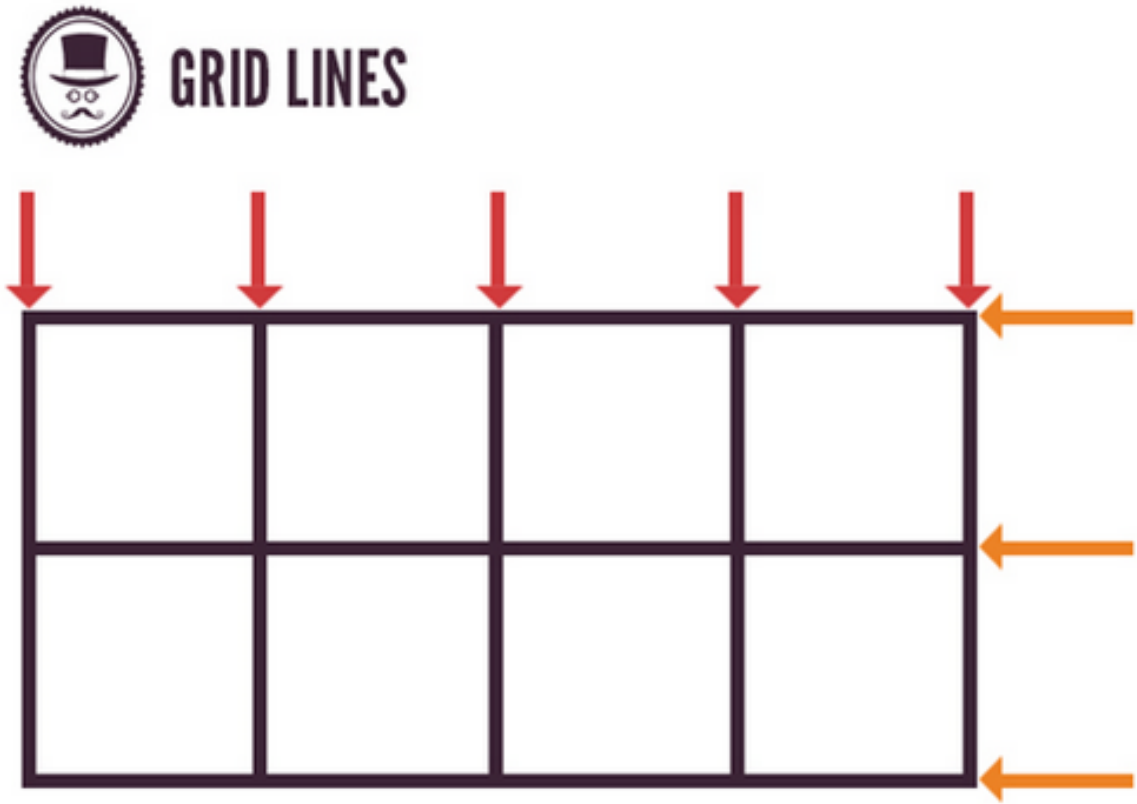
- Assim como no Flexbox, vamos dividir as propriedades do GridLayout em propriedades do **container** (elemento pai do layout) e propriedades dos **grid-itens** (itens desse container);
- O GridLayout é bem extenso, portanto vamos cobrir apenas o básico das suas propriedades (você pode consultar mais em: https://css-tricks.com/snippets/css/complete-guide-grid)
- As definições da GridLayout são:
  - **Grid**: área que visualmente possui linhas e colunas (como se fosse tabela);
  - **Grid Line**: linhas divisórias que estruturam o grid;
  - **Grid Item**: item interno da grid (qualquer tag HTML);
  - **Grid Track**: espaço entre as linhas (vertical ou horizontal);
  - **Grid Area**: qualquer espaço entre quatro linhas dentro de um grid.

Grid Layout



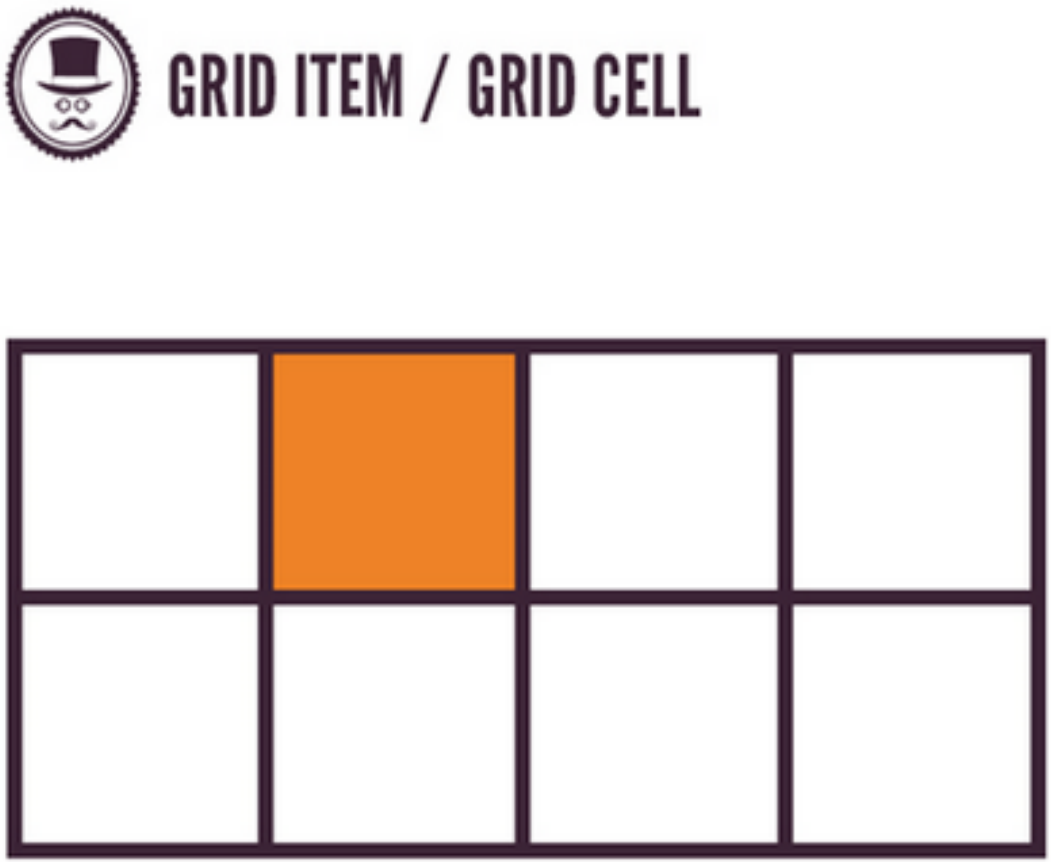
- Créditos das imagens: http://www.chiefofdesign.com.br/css-grid-layout

Grid Layout



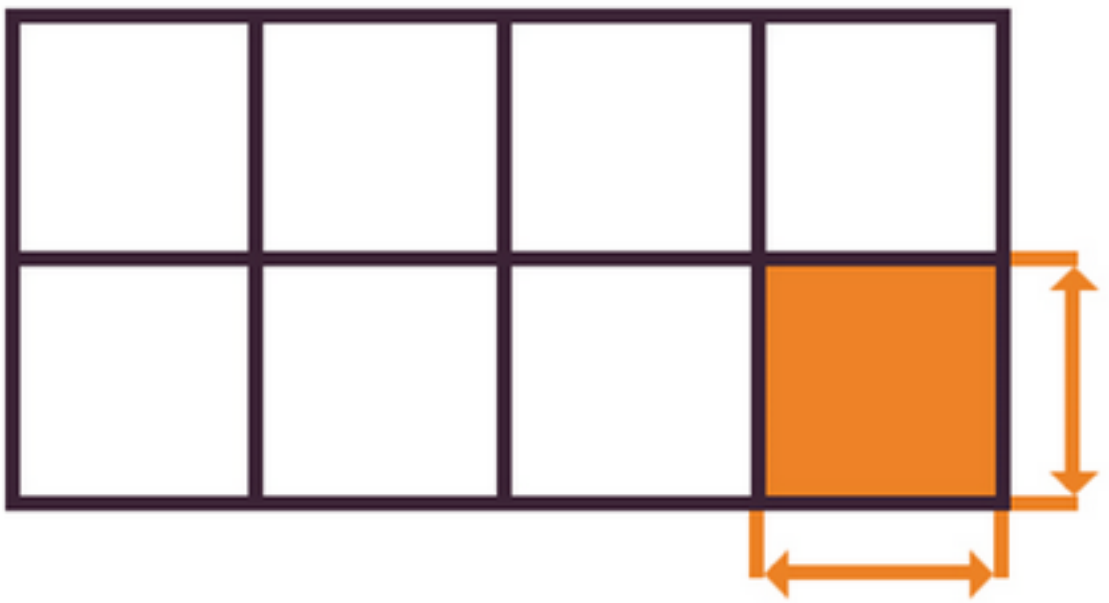
- Créditos das imagens: http://www.chiefofdesign.com.br/css-grid-layout

Grid Layout



- Créditos das imagens: http://www.chiefofdesign.com.br/css-grid-layout

Grid Layout



- Créditos das imagens: http://www.chiefofdesign.com.br/css-grid-layout

Grid Layout

- Créditos das imagens: http://www.chiefofdesign.com.br/css-grid-layout

Grid Layout

- Dentro das definições do Grid, temos as seguintes propriedades para o **container**:
  - **Grid-template-columns**: Define quantas colunas e o tamanho básico de cada uma;
  - **Grid-template-rows**: Define quantas linhas e o tamanho básico de cada;
  - **Grid-template-areas**: Define área nomeadas por classe do grid;
  - **Grid-row-gap**: Espaço entre linhas;
  - **Grid-column-gap**: Espaço entre colunas.

Grid Layout

- Dentro das definições do Grid, temos as seguintes propriedades para os **itens**:
  - `Grid-column|row-start`: Determina em qual coluna ou linha o item começa;
  - `Grid-column|row-end`: Determina em qual coluna ou linha o item termina;
  - `Grid-area`: Determina qual área nomeada (classe) do grid esse item vai ocupar;
- Para entender mais sobre o grid, veja: http://cssgridgarden.com

Conteúdo

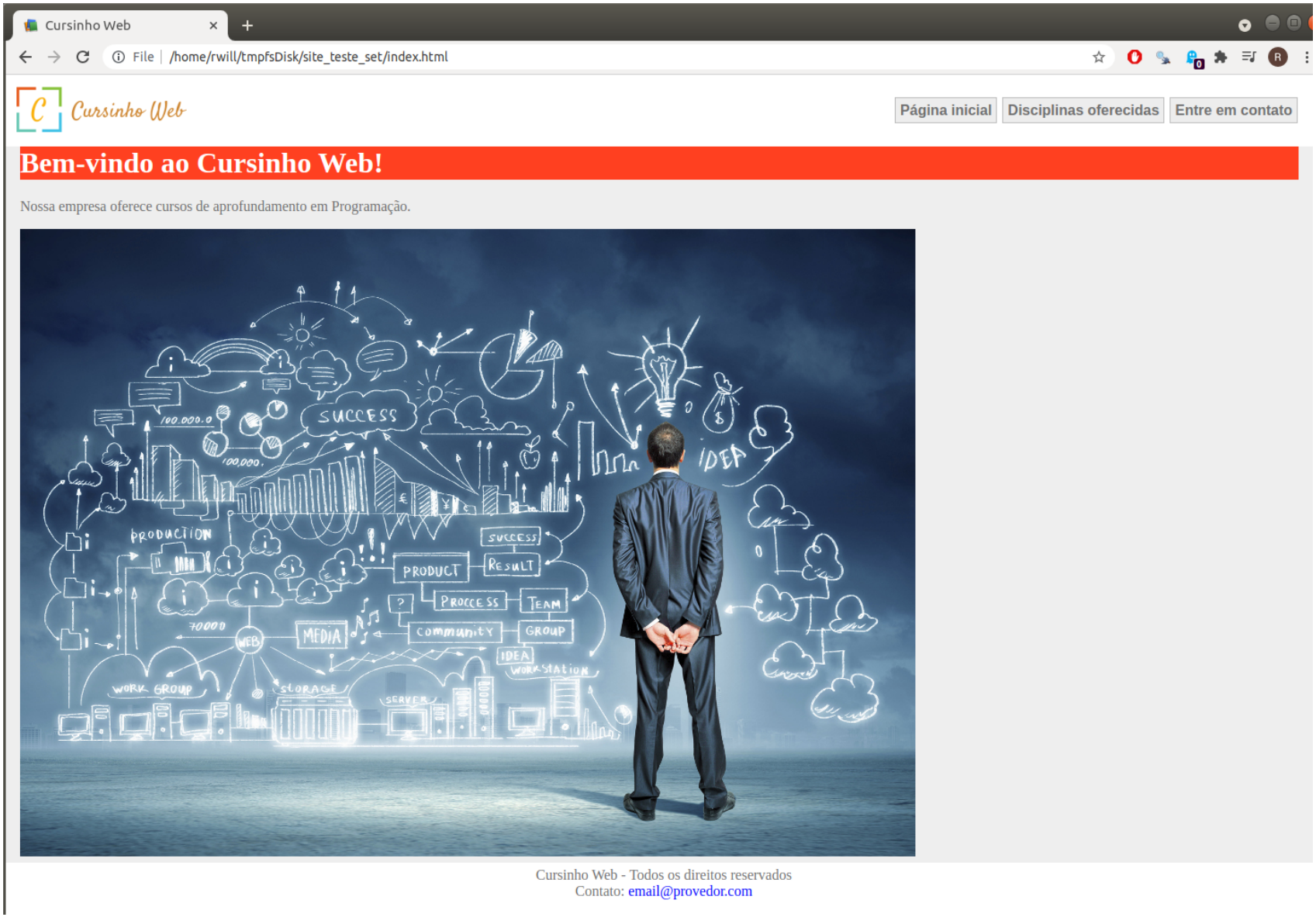
Exercício

- Baixe o arquivo com o logotipo da empresa fictícia "Cursinho Web" e outras imagens atualizadas com fundo transparente. Em seguida, faça:
  - Adicione a imagem do logotipo ao início do *header* com o *id* "img-logo" e remova as barras verticais entre os links dentro do nav. Isso deve ser replicado em todas as páginas HTML.
  - Adicione a propriedade `margin` com valor 0px no seletor do *body*.
  - Adicione as seguintes propriedades ao seletor *header*:
    - Altura (`height`) 80px;
    - Margem (`margin`) 0px;
    - Padding (`padding-right`) à direita de 1rem;
    - *display* flex;
    - `justify-content` com valor *space-between*;
    - `align-items` com valor *center*;
  - Adicione um seletor para a tag *main* com os seguintes valores:
    - Margem esquerda: 1rem;
    - Margem direita: 1rem;
    - Margem do topo: 0px;
    - Margem do fundo: 100px;
  - Adicione as seguintes propriedades ao seletor *footer*:
    - Alinhamento do texto: centralizado;
    - Comprimento (`width`) 100%;
    - Propriedade `position` com o valor *fixed* (força o *footer* a ficar num lugar fixo, ao invés de seguir a sequencia de tags da página);
    - Propriedade `bottom` com valor 0 (força o *footer* a ficar preso na parte de baixo da página).
    - Padding de 5px;
  - Mude a margem do topo para 0px no seletor *h1*.

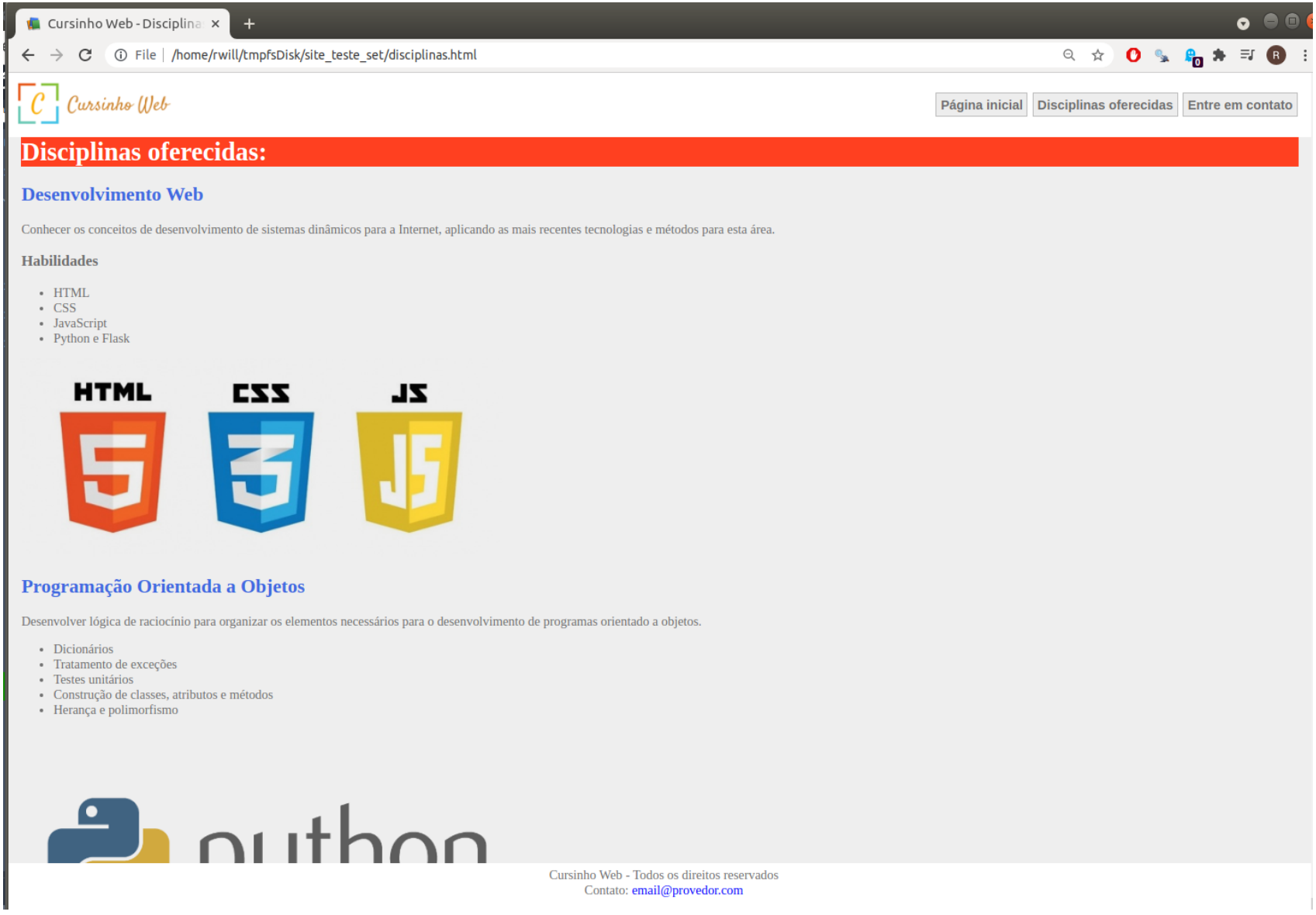
Exercício

- (continuação):
  - Adicione as seguintes propriedades a todos os links dentro do *nav*:
    - Cor de fundo: #ededed;
    - Borda: 1px, borda sólida com cor #bbbbbb;
    - Padding de 5px;
    - Margem: 1px;

Exercício: *index.html*



Exercício: *disciplinas.html*



Exercício: *contato.html*