

WEB DEVELOPMENT – PARTE 1

CAIXAS

ISRAEL MARQUES CAJAI JUNIOR



4

LISTA DE FIGURAS

Figura 4.1 - Box Model	5
Figura 4.2 – Margens externas (margin)	5
Figura 4.3 – Bordas (border)	7
Figura 4.4 – Bordas sendo exibidas no navegador	8
Figura 4.5 – Tipos de bordas.....	9
Figura 4.6 – Padding (preenchimento)	10
Figura 4.7 – Parágrafos com padding definido.....	11
Figura 4.8 – Organização de layout com divs - Wireframe.....	12
Figura 4.9 – Representação dos elementos block e inline	13
Figura 4.10 – Exibição dos elementos em bloco	14
Figura 4.11 – Exibição dos elementos em linha	15
Figura 4.12 – Cabeçalho da página	18
Figura 4.13 – Exibição na página das duas divs	19
Figura 4.14 – Exibição da página com três divs	20
Figura 4.15 – Rodapé da página	21
Figura 4.16 – Parte da página do Gog.com para o game Cuphead	23

LISTA DE CÓDIGOS-FONTE

Código-fonte 4.1 – Definindo um valor único para as margens.....	6
Código-fonte 4.2 – Definindo valores individuais das margens	6
Código-fonte 4.3 – Definindo valores individuais em uma única linha.....	6
Código-fonte 4.4 – Definindo dois valores para as margens	6
Código-fonte 4.5 – Definindo três valores para as margens.....	7
Código-fonte 4.6 – Criando parágrafos com HTML e formatando com CSS	8
Código-fonte 4.7 – Regra CSS para borda simplificada	9
Código-fonte 4.8 – Atribuindo padding aos parágrafos	10
Código-fonte 4.9 – Criando uma div	13
Código-fonte 4.10 – Exemplo de elementos em bloco	14
Código-fonte 4.11 – Exemplo de elementos em linha	15
Código-fonte 4.12 – Código HTML - criando o container “tudo”	16
Código-fonte 4.13 – Código CSS para formatar o id “tudo”	17
Código-fonte 4.14 – Código HTML para o id “topo”	17
Código-fonte 4.15 – Código CSS para o id “topo”	18
Código-fonte 4.16 – Código HTML com duas divs	18
Código-fonte 4.17 – Código CSS para as duas divs	19
Código-fonte 4.18 – Código HTML para as três divs.....	20
Código-fonte 4.19 – Código CSS para as três divs	20
Código-fonte 4.20 – Código HTML para o id “rodape”	21
Código-fonte 4.21 – Código CSS para o id “rodapé”	21

SUMÁRIO

4 CAIXAS	5
4.1 Margin	5
4.2 Border.....	7
4.3 Padding	9
4.4 Criando caixas - DIV.....	11
4.5 Alinhamento de elementos html	13
4.6 Codificando o wireframe.....	15
4.6.1 Criando o container principal	16
4.6.2 Criando o container de cabeçalho.....	17
4.6.3 Criando a divisão em duas divs.....	18
4.6.4 Criando a divisão em três divs	20
4.6.5 Criando o container de rodapé	21
4.7 Considerações sobre caixas	21
REFERÊNCIAS.....	24

4 CAIXAS

Imagine que todos os elementos HTML possuam uma “caixa invisível” em sua volta, elas são usadas para delimitar o nosso conteúdo, a isso chamamos de Box Model, composto por: margens externas (margin), bordas (border), margens internas ou preenchimento (padding) e conteúdo (content).



Figura 4.1 - Box Model
Fonte: Elaborado pelo autor (2017)

4.1 Margin

Os valores atribuídos à margin poderão definir a distância que uma caixa terá de outras ou das extremidades da janela do browser.

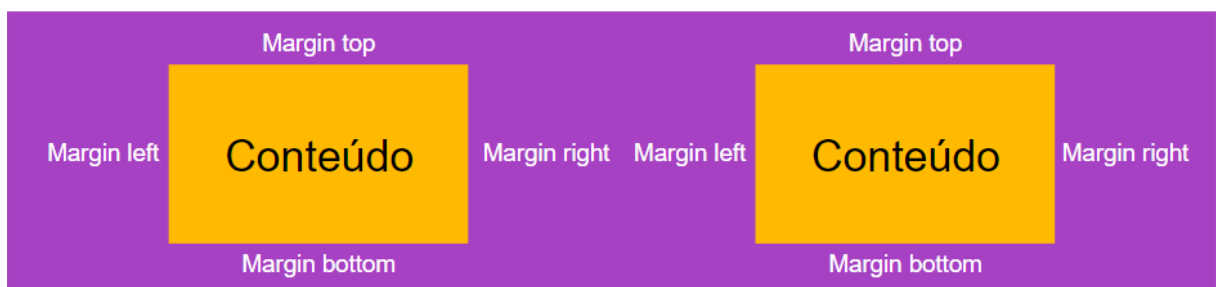


Figura 4.2 – Margens externas (margin)
Fonte: Elaborado pelo autor (2017)

Podemos definir um único valor que será usado por todas as margens top, right, bottom e left, ou valores individuais para cada uma.

```
h1{  
    margin: 20px;  
}
```

Código-fonte 4.1 – Definindo um valor único para as margens
Fonte: Elaborado pelo autor (2017)

Podemos também definir valores diferentes para cada uma das margens de algum elemento, no exemplo, usaremos o h1.

```
h1{  
    margin-top: 20px;  
    margin-right: 30px;  
    margin-bottom: 40px;  
    margin-left: 50px;  
}
```

Código-fonte 4.2 – Definindo valores individuais das margens
Fonte: Elaborado pelo autor (2017)

Podemos também fazer essa declaração em uma única linha, seguindo esta ordem de valores: top, right, bottom, left

```
h1{  
    margin: 20px 30px 40px 50px;  
}
```

Código-fonte 4.3 – Definindo valores individuais em uma única linha
Fonte: Elaborado pelo autor (2017)

Pode ser feita também a declaração de apenas dois valores, dessa forma, o browser entenderá que o primeiro valor será aplicado para as margens top e bottom e o segundo para right e left.

```
h1{  
    margin: 20px 30px;  
}
```

Código-fonte 4.4 – Definindo dois valores para as margens
Fonte: Elaborado pelo autor (2017)

Também é possível fazer a declaração de três valores, dessa forma o browser entenderá que o primeiro valor será aplicado na margem top, o segundo para right e left, e o terceiro na margem bottom.

```
h1{
```

```
margin: 20px 25px 30px;  
}
```

Código-fonte 4.5 – Definindo três valores para as margens
Fonte: Elaborado pelo autor (2017)

Essas formas diferentes de representar valores também são válidas para as bordas (border) e para as margens internas (padding).

4.2 Border

Os valores atribuídos ao border definirão se a caixa possui ou não um contorno que ficará em volta do preenchimento e do conteúdo. Podemos também estabelecer a largura desse contorno, seu estilo e a sua cor.

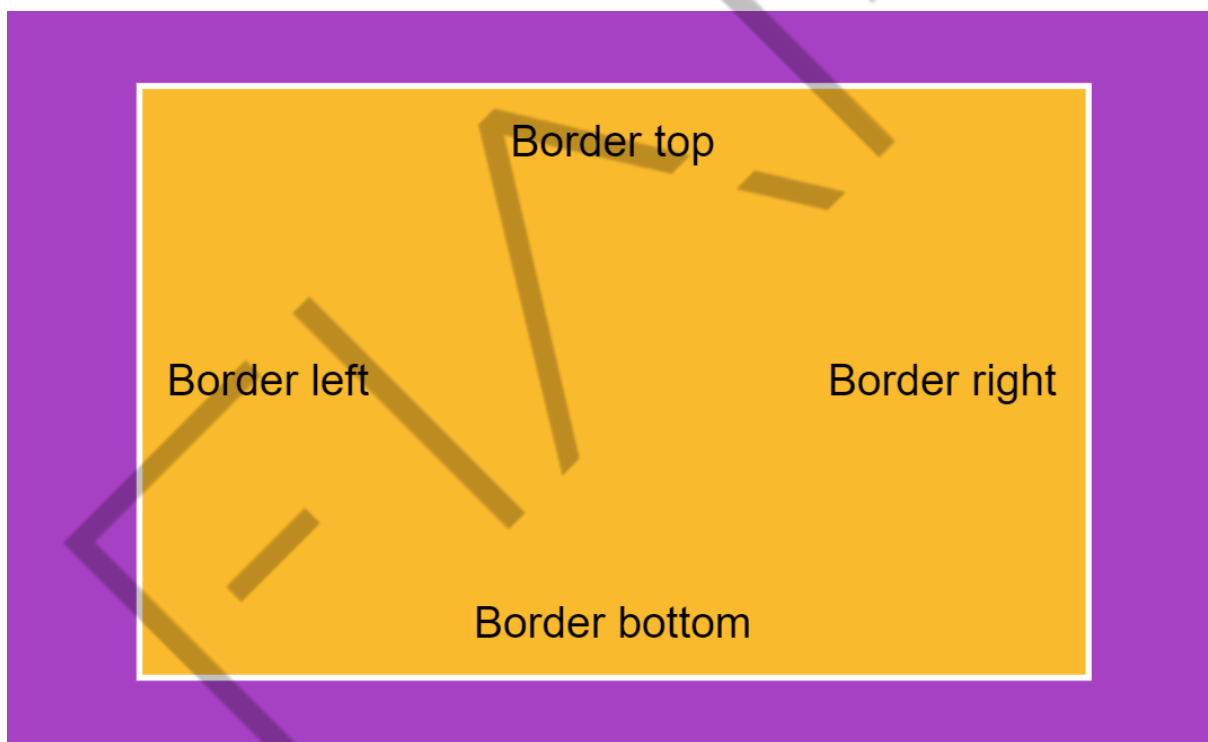


Figura 4.3 – Bordas (border)
Fonte: Elaborado pelo autor (2017)

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Escape Adventure</title>  
    <style type="text/css">  
      p{  
        border-width: 5px;  
        border-style: solid;
```

```
        border-color: #cc0033;
    }
</style>
</head>
<body>
    <p>Escape Adventure</p>
    <p>Escape Adventure</p>
    <p>Escape Adventure</p>
</body>
</html>
```

Código-fonte 4.6 – Criando parágrafos com HTML e formatando com CSS

Fonte: Elaborado pelo autor (2017)

No código-fonte “Criando parágrafos com HTML e formatando com CSS”, temos a marcação HTML, fizemos uso da tag <p> para a criação dos três parágrafos com o mesmo conteúdo. Foi definida uma regra CSS interna que estabeleceu a borda dos elementos como: largura em 5px, estilo sólido e código hexadecimal para a cor #CC0033. Dessa forma, **todos os parágrafos** da página terão sempre a mesma formatação.

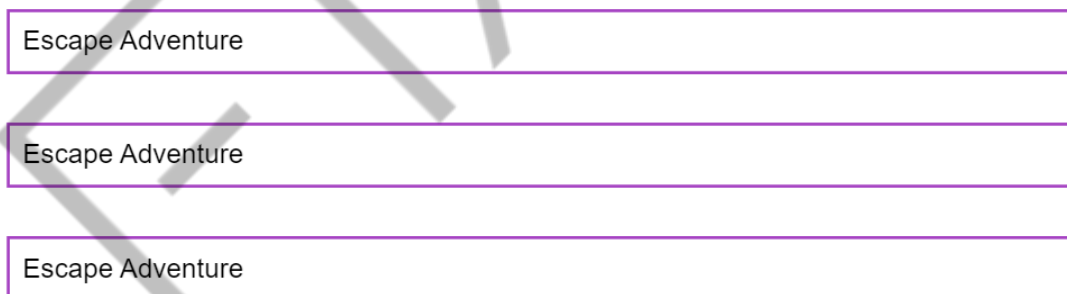


Figura 4.4 – Bordas sendo exibidas no navegador

Fonte: Elaborado pelo autor (2017)

Podemos diminuir a quantidade de linhas da regra CSS fazendo a marcação simplificada, citando as propriedades em uma única linha.

```
p{
    border: 5px solid #cc0033;
}
```


Código-fonte 4.7 – Regra CSS para borda simplificada
Fonte: Elaborado pelo autor (2017)

A CSS permite os seguintes estilos de bordas: dotted, dashed, solid, double, groove, ridge, inset, outset.

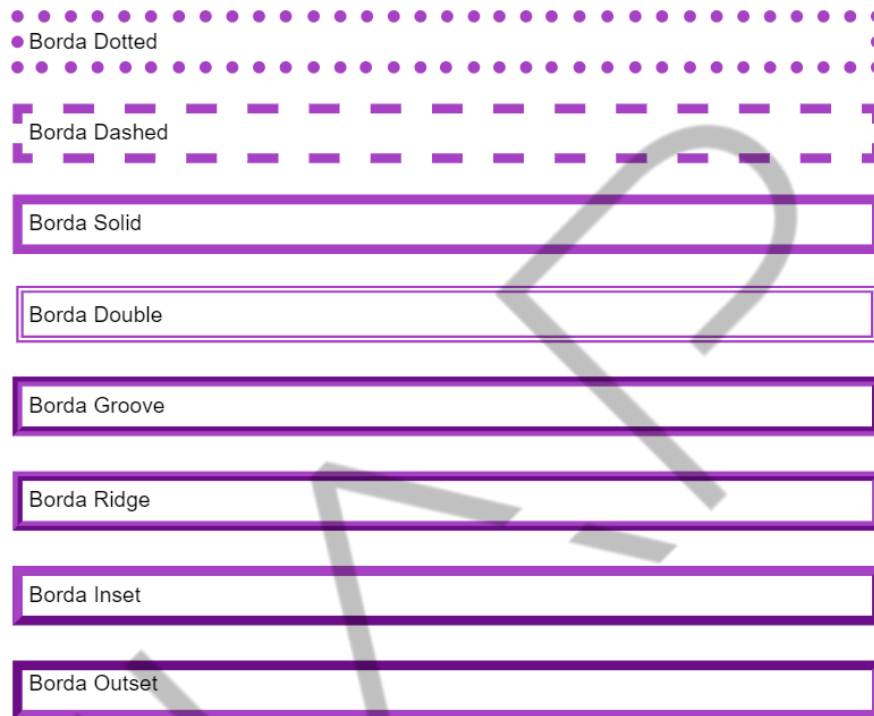


Figura 4.5 – Tipos de bordas
Fonte: Elaborado pelo autor (2017)

4.3 Padding

Os valores atribuídos ao padding definirão a distância que o conteúdo terá das extremidades internas (preenchimento) da caixa.

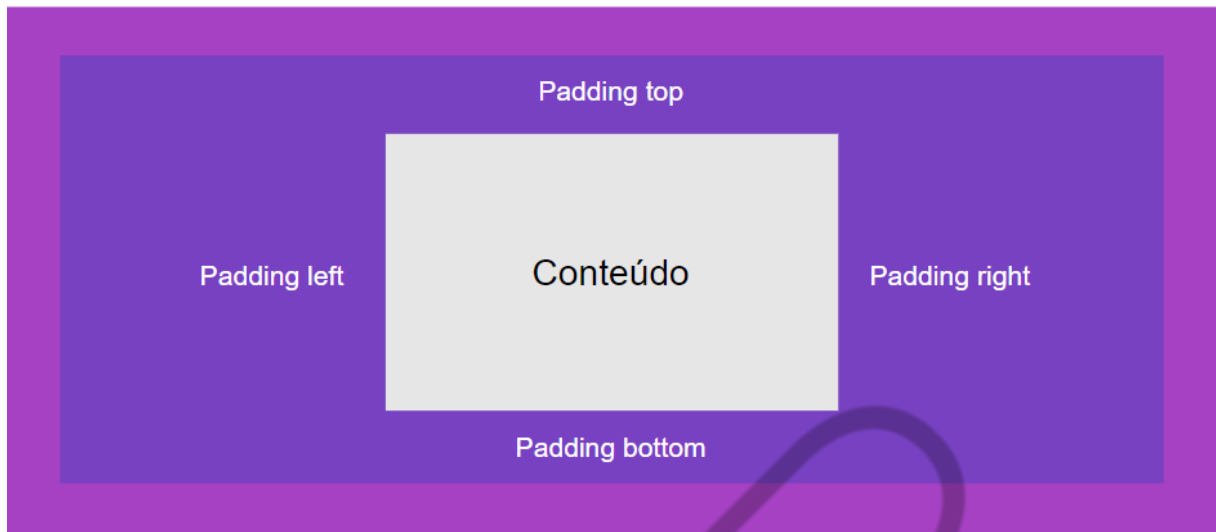


Figura 4.6 – Padding (preenchimento)
Fonte: Elaborado pelo autor (2017)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Escape Adventure</title>
    <style type="text/css">
      p{
        border-width: 5px;
        border-style: solid;
        border-color: #cc0033;
        padding: 20px;
      }
    </style>
  </head>
  <body>
    <p>Escape Adventure</p>
    <p>Escape Adventure</p>
    <p>Escape Adventure</p>
  </body>
</html>
```

Código-fonte 4.8 – Atribuindo padding aos parágrafos
Fonte: Elaborado pelo autor (2017)

Na regra CSS do código-fonte “Atribuindo padding aos parágrafos”, adicionamos o valor de 20px ao padding (preenchimento) dos parágrafos, sendo assim, o conteúdo ficará afastado das bordas.

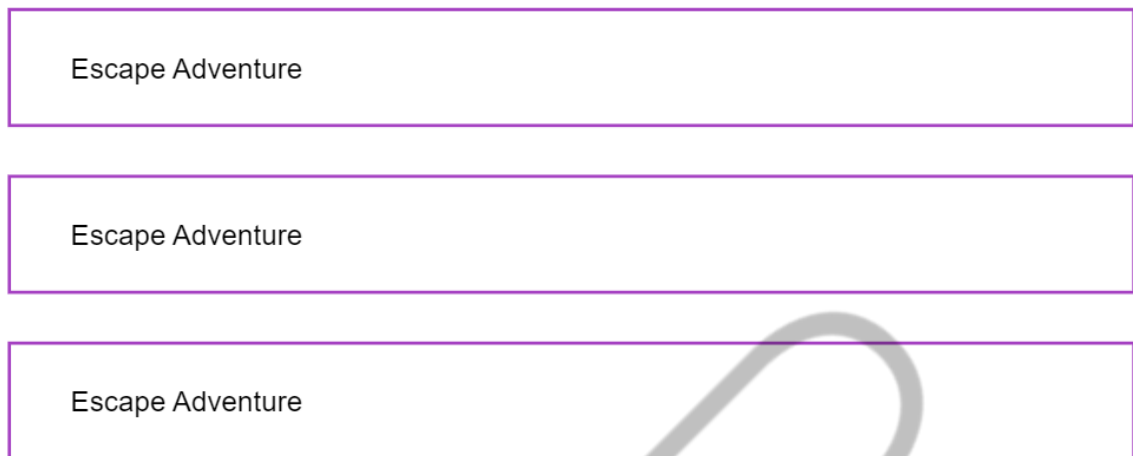


Figura 4.7 – Parágrafos com padding definido
Fonte: Elaborado pelo autor (2017)

4.4 Criando caixas - DIV

A tag `<div>` agrupará um determinado conteúdo dentro de uma caixa (box) e, por padrão, não permitirá que outros elementos fiquem ao seu lado. Dentro desse box podemos inserir qualquer elemento HTML, qualquer tipo de conteúdo e até mesmo caixas dentro de outras caixas. Use essa tag para agrupar partes distintas da sua página, as chamadas seções, pois por meio dela podemos organizar todo o layout de nosso site. Observe a figura “Organização de layout com divs – Wireframe”, que representa o desenho de uma página web.

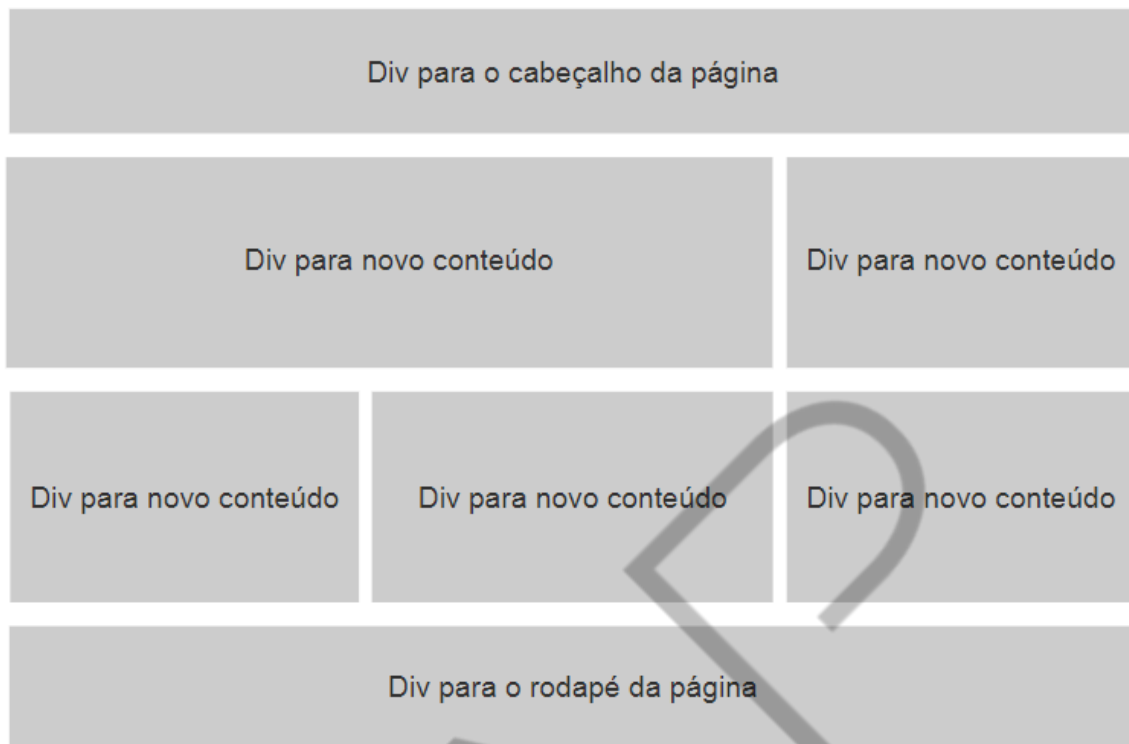


Figura 4.8 – Organização de layout com divs - Wireframe
Fonte: Elaborado pelo autor (2017)

Foi desenhado um protótipo (wireframe) que mostra como deverá ser apresentado o conteúdo de uma determinada página web, separado em várias divs. A criação de um wireframe ajudará no desenvolvimento do código HTML e CSS. É muito importante que faça o planejamento de seu site antes de começar a codificação. Organize-se:

- Separe todo o conteúdo que deseja inserir nas páginas: textos, imagens, vídeos, tabelas, formulários etc.
- Escolha uma paleta de cores, existem ótimos gerenciadores on-line que podem ajudá-lo, um deles é o Adobe Color CC (<https://color.adobe.com>).
- Estabeleça as fontes que serão usadas, como falamos anteriormente, use um servidor de fontes.
- Faça o tratamento das imagens, definindo um padrão de tamanho e cores. Existem vários programas que fazem essa tarefa. Adobe Fireworks (<http://www.adobe.com/br/products/fireworks.html>) e Adobe Photoshop, (<https://www.adobe.com/br/products/photoshop.html>), são excelentes ferramentas.

- Desenhe os wireframes de todas as páginas, por meio deles terá uma ótima ideia de como seu site ficará. Devem representar fielmente a estruturação do layout e hierarquização do conteúdo. Na Internet, pode utilizar o Moqups (<https://moqups.com/>), Wireframe CC, (<https://wireframe.cc/>), que é bem mais simples e não possui tantos recursos como o Moqups, ou então baixar um software específico, como o Axure (<https://www.axure.com/>).

```
<div>
  <h1>Escape Adventure</h1>
  <p>A sua aventura está apenas começando.</p>
</div>
```

Código-fonte 4.9 – Criando uma div
Fonte: Elaborado pelo autor (2017)

4.5 Alinhamento de elementos html

Podemos definir como as caixas (boxes) ficarão alinhadas em relação à janela do browser, algumas permitem que outras caixas fiquem ao seu lado (inline), enquanto outras ocupam todo o espaço não permitindo que nenhuma fique ao seu lado (block).

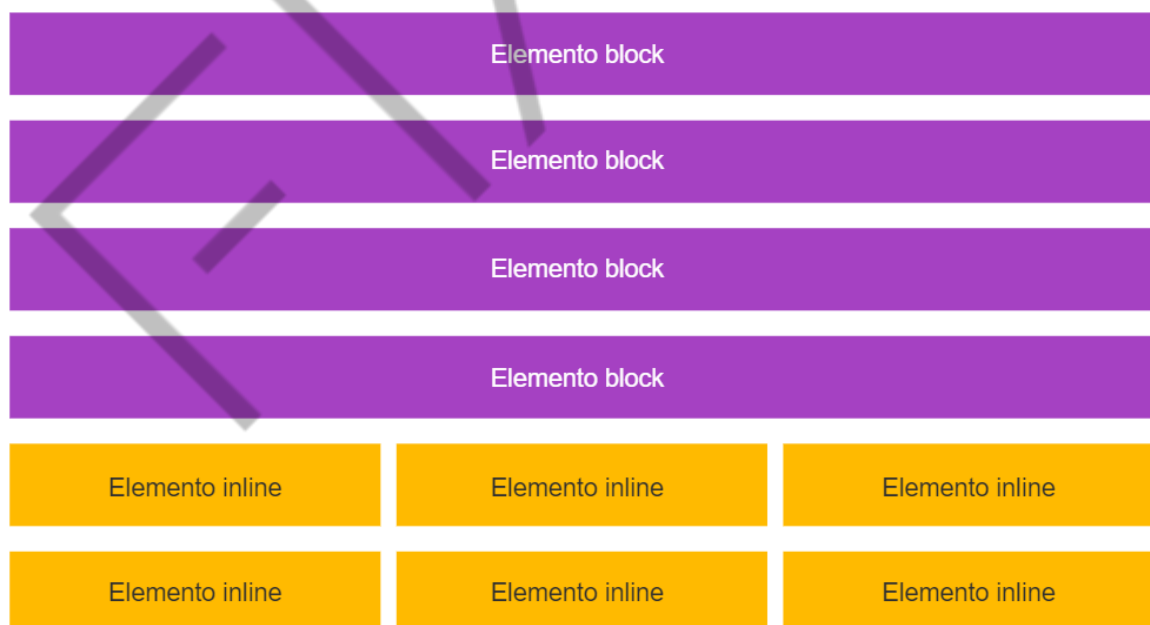


Figura 4.9 – Representação dos elementos block e inline
Fonte: Elaborado pelo autor (2017)

Por padrão, os elementos em bloco ocupam sempre uma nova linha na janela do browser. As seguintes tags possuem o elemento block: <h1>, <h2>, <h3>, <p>,

, . Vale ressaltar que, com a CSS, podemos mudar esse comportamento do elemento.

```
<h1>Escape Adventure</h1>
<p>A sua aventura está apenas começando.</p>
<ul>
  <li>História</li>
  <li>Personagens</li>
  <li>Fases</li>
  <li>Demo</li>
</ul>
```

Código-fonte 4.10 – Exemplo de elementos em bloco
Fonte: Elaborado pelo autor (2017)



Figura 4.10 – Exibição dos elementos em bloco
Fonte: Elaborado pelo autor (2017)

Os elementos inline permitirão que outras caixas ocupem a mesma linha, ficando uma ao lado da outra. As seguintes tags possuem o elemento inline: , <a>, , <input>, <label>, <select>. Vale ressaltar que, com a CSS, podemos mudar esse comportamento do elemento.

```
<body>
  
  
  
```

```
</body>
```

Código-fonte 4.11 – Exemplo de elementos em linha
Fonte: Elaborado pelo autor (2017)

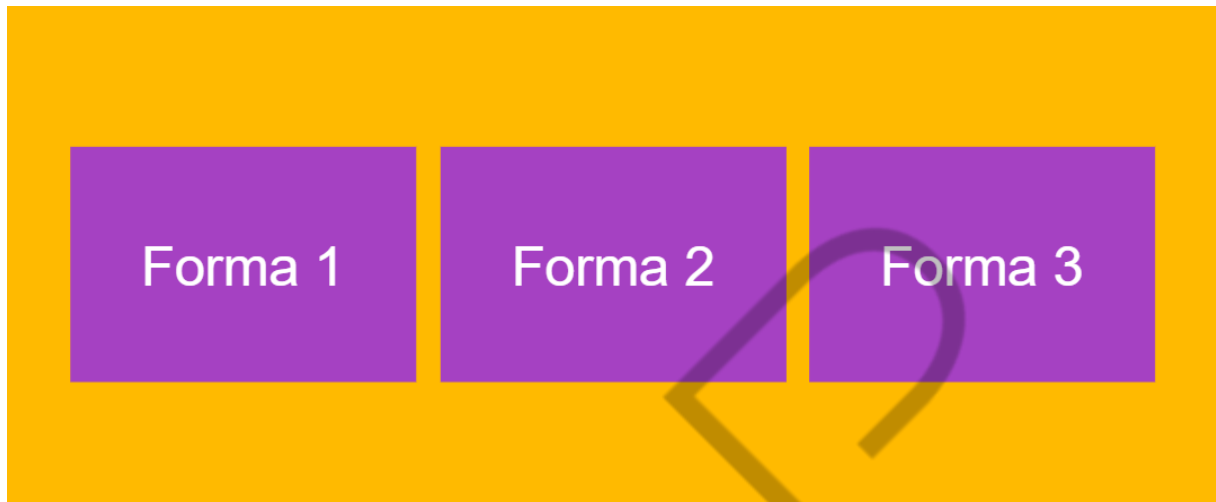


Figura 4.11 – Exibição dos elementos em linha
Fonte: Elaborado pelo autor (2017)

Existe também o elemento `inline-block`, em que uma caixa é posicionada inline, ficando na mesma linha na janela do browser, mas comporta-se como se fosse um elemento em bloco, usaremos bastante quando criarmos os menus de navegação.

4.6 Codificando o wireframe

Agora que sabemos o que é um box, como é formado e como se comporta em relação à linha do navegador, vamos codificar o wireframe apresentado na figura “Organização de layout com divs”, a marcação HTML e a formatação CSS serão separadas para que se possa entender melhor cada parte do código.

Como o nosso protótipo é composto por várias tags `<div>`, iremos usar o atributo `id`, que servirá para identificar cada um dos elementos HTML. O nome do identificador deve começar sempre com letras ou underline, nunca com números ou outro caractere especial. Teremos de tomar um cuidado todo especial, pois, em uma página, não poderemos ter dois ou mais elementos com o mesmo `id`, ou seja, deve ser único.

Quando definimos uma identidade exclusiva para um elemento, possibilitamos que seja formatado de forma diferente dos demais elementos iguais a ele. Imagine que em nossa página tenhamos quatro elementos `<h2>`, quando montarmos a regra

CSS para todos, ficarão iguais. Agora, se colocarmos um id em algum deles, poderemos criar uma formatação exclusiva que afetará somente aquele elemento que possuir o id.

Quando formos aprender JavaScript, o id será muito importante, pois podemos pegar informações de um determinado elemento por meio dele. Se tivermos dois ids iguais, poderemos ter um erro nessa manipulação. Lembre-se: id é único na página.

4.6.1 Criando o container principal

Para facilitar a marcação, criamos uma caixa principal que receberá todas as outras divs, ela terá um tamanho fixo e ficará centralizada na tela. A ideia é deixar o conteúdo dentro de um container, facilitando assim a sua formatação e o seu posicionamento. Esse elemento recebeu como id a palavra “tudo”.

```
<div id="tudo">
  <!-- Aqui vem todo o conteúdo da página -->
</div>
```

Código-fonte 4.12 – Código HTML - criando o container “tudo”
Fonte: Elaborado pelo autor (2017)

Criamos o elemento **<div>** representado pelo id “tudo”, será dentro desse elemento que colocaremos todas as outras divs e os seus respectivos conteúdos. Repare que existe a seguinte marcação no código: **<!-- Aqui vem todo o conteúdo da página -->**, isso representa um comentário dentro do código HTML que não será exibido pelo navegador quando a página for carregada. Use sempre esse recurso, comentar blocos de código facilita a sua leitura e posterior manutenção. No código CSS, comentários são representados pela marcação **/* COMENTÁRIOS */**.

Na CSS começaremos com uma mudança, em vez de definir como seletor a tag **<div>**, usaremos o sinal de cerquilha (**#**) junto da palavra “tudo”. Será entendido pelo navegador que está sendo feita uma referência a um id específico, então ele procurará esse elemento no código HTML e aplicará a regra CSS. **Será sempre dessa forma que representaremos os ids dentro do código CSS.**

O elemento terá largura em 800px, altura em 640px, margem externa superior e inferior em 10px, margem externa direita e esquerda automáticas, isso fará com que

a <div> sempre fique centralizada na janela do navegador. Terá também borda de 1px de largura, no estilo sólido (linha contínua) e cor preta.

```
#tudo{
  width: 800px;
  height: 640px;
  margin: 10px auto;
  border: 1px solid #000;}
```

Código-fonte 4.13 – Código CSS para formatar o id “tudo”
Fonte: Elaborado pelo autor (2017)

4.6.2 Criando o container de cabeçalho

Dentro do elemento “tudo”, utilizamos a tag <div> para criar um container que recebeu como id a palavra “topo”. Em seu interior foi inserido um elemento <h1> como a frase “Div para o cabeçalho da página”, nunca se esqueça de fazer o fechamento de qualquer <div>.

```
<div id="tudo">
  <div id="topo">
    <h1>Div para o cabeçalho da página</h1>
  </div>
</div>
```

Código-fonte 4.14 – Código HTML para o id “topo”
Fonte: Elaborado pelo autor (2017)

Na regra CSS definimos que o elemento terá a cor de fundo cinza, largura em 700px, altura em 80px, margem externa superior e inferior em 10px, margem externa direita e esquerda automáticas, isso fará com que a div sempre fique centralizada em relação ao seu elemento pai, nesse caso, o id “tudo”. Terá também borda de 1px de largura, no estilo sólido e cor preta. Também foi feita uma regra para o elemento <h1> que está dentro do id topo, em que seu texto será centralizado. Observe que a marcação foi bem específica, essa regra CSS somente será aplicada ao elemento <h1> que estiver dentro do id “topo”. Se na página existir outra tag <h1> fora desse id, essa regra não será aplicada.

```
#topo{
  background: #ccc;
  width: 780px;
  height: 80px;
  margin: 10px auto;
  border: 1px solid #000;}
```

```
#topo h1{  
  text-align: center;}
```

Código-fonte 4.15 – Código CSS para o id “topo”
Fonte: Elaborado pelo autor (2017)

Div para o cabeçalho da página

Figura 4.12 – Cabeçalho da página
Fonte: Elaborado pelo autor (2017)

4.6.3 Criando a divisão em duas divs

Dentro do elemento “tudo”, utilizamos a tag <div> para criar dois containers, “coluna-um” e “coluna-dois”. Em seus interiores foi inserido um elemento <h3> como a frase “Div para novo conteúdo”.

```
<div id="coluna-um">  
  <h3>Div para novo conteúdo</h3>  
</div>  
  
<div id="coluna-dois">  
  <h3>Div para novo conteúdo</h3>  
</div>
```

Código-fonte 4.16 – Código HTML com duas divs
Fonte: Elaborado pelo autor (2017)

A regra CSS basicamente é a mesma da anterior, mas a sua chamada mudará um pouco. Como temos dois elementos que possuem as mesmas formatações, vamos referenciar os ids juntos, separando-os com vírgula (,). Dessa forma, o navegador entenderá que a regra deve ser aplicada aos dois elementos.

Observe, no código-fonte “CSS para as duas divs”, que alteramos a largura das divs para 378px, pois elas estão dentro de um elemento pai que tem largura máxima de 800px. Levando em consideração que os elementos possuem borda e margem, então precisamos fazer uma conta básica para definir sua largura e ter a certeza de que sempre caibam no container: **larguras dos elementos + margens + bordas = tamanho total**. No código, temos: $(378 \times 2) + 40 + 4 = 800\text{px}$. Cada elemento tem largura de 378px, possui 20px de margem esquerda e direita e 2px de borda esquerda e direita.

Existe também no código a propriedade float, que determina que um elemento deve flutuar dentro do seu container, que utilizamos para definir o posicionamento e o layout em páginas da web. No caso, os dois elementos foram definidos com uma flutuação à esquerda, fazendo com que fiquem um ao lado do outro. Abaixo, é encontrada uma nova regra para o id “coluna-dois”, definindo apenas a flutuação como à direita. O navegador irá ignorar a primeira regra e atribuir esta nova, permanecendo ao lado do id “coluna-um”, mas flutuando à direita do elemento pai.

Também foi feita uma regra para os elementos <h3> que estão dentro dos ids “coluna-um” e “coluna-dois”, deixando o texto centralizado. Observe que a marcação foi bem específica, essa regra CSS somente será aplicada aos elementos <h3> que estiverem dentro dos ids respectivos. Se na página existir outra tag <h3> fora desses ids, essa regra não será aplicada.

```
#coluna-um, #coluna-dois{
    background: #ccc;
    width: 378px;
    height: 200px;
    border: 1px solid #000;
    margin: 10px;
    float: left;}

#coluna-dois{
    float:right;}

#coluna-um h3, #coluna-dois h3{
    text-align: center;}
```

Código-fonte 4.17 – Código CSS para as duas divs
Fonte: Elaborado pelo autor (2017)



Figura 4.13 – Exibição na página das duas divs
Fonte: Elaborado pelo autor (2017)

4.6.4 Criando a divisão em três divs

Dentro do elemento “tudo” e abaixo das duas colunas, utilizamos a tag <div> para criar três containers, “parte-um”, “parte-dois” e “parte-tres”. Em seus interiores foi inserido um elemento <h3> como a frase “Div para novo conteúdo”.

```
<div id="parte-um">
  <h3>Div para novo conteúdo</h3>
</div>

<div id="parte-dois">
  <h3>Div para novo conteúdo</h3>
</div>

<div id="parte-tres">
  <h3>Div para novo conteúdo</h3>
</div>
```

Código-fonte 4.18 – Código HTML para as três divs
Fonte: Elaborado pelo autor (2017)

Utilizamos a mesma regra CSS do código-fonte “CSS para as duas divs”, diminuindo apenas a largura dos ids para 244px, com a flutuação à esquerda. Note que também deixamos a propriedade text-align dentro da regra, assim, automaticamente centralizará os seus elementos.

```
#parte-um, #parte-dois, #parte-tres{
  float: left;
  background: #ccc;
  width: 244px;
  height: 200px;
  border: 1px solid #000;
  margin: 10px;
  text-align: center;}

```

Código-fonte 4.19 – Código CSS para as três divs
Fonte: Elaborado pelo autor (2017)

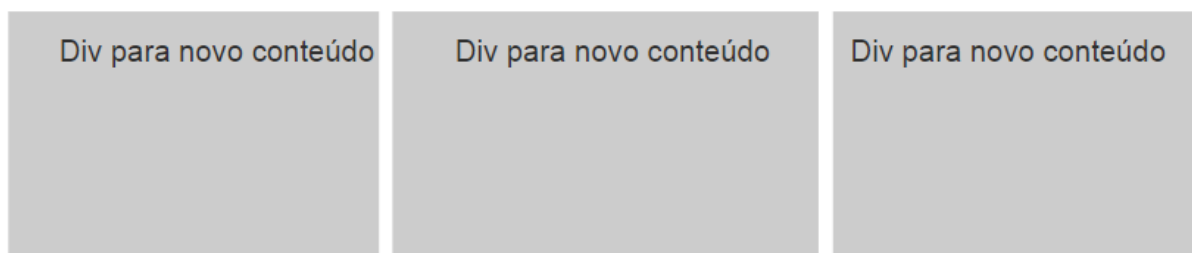


Figura 4.14 – Exibição da página com três divs
Fonte: Elaborado pelo autor (2017)

4.6.5 Criando o container de rodapé

Será idêntico ao container de cabeçalho, com mudança apenas no nome do id que passará a ser chamado de “rodapé”.

```
<div id="rodape">
  <h3>Div para o rodapé da página</h3>
</div>
</div><!--Fechamento da div tudo -->
```

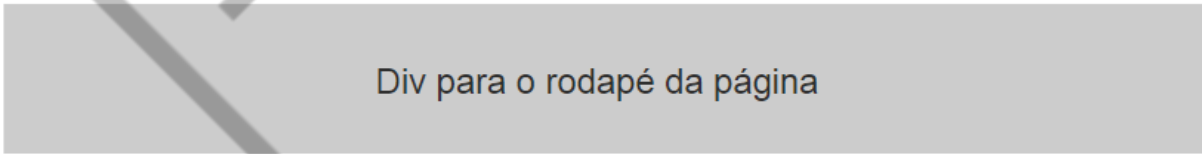
Código-fonte 4.20 – Código HTML para o id “rodapé”
Fonte: Elaborado pelo autor (2017)

No CSS inserimos a propriedade clear, que irá determinar se os elementos podem flutuar e de qual lado isso pode ocorrer. Às vezes, quando usamos a propriedade float, pode acontecer de o elemento imediatamente posterior a outro elemento ficar fora de posicionamento, então utilizamos a propriedade clear para limpar essa flutuação. Os valores possíveis são: left, right ou both.

```
#rodape{
  clear: both;
  background: #ccc;
  width: 780px;
  height: 80px;
  margin: 10px auto;
  border: 1px solid #000;
  text-align: center;}

```

Código-fonte 4.21 – Código CSS para o id “rodapé”
Fonte: Elaborado pelo autor (2017)



Div para o rodapé da página

Figura 4.15 – Rodapé da página
Fonte: Elaborado pelo autor (2017)

4.7 Considerações sobre caixas

É muito importante que você entenda esses tópicos, pois aqui temos uma primeira base de como podemos criar um layout para uma página HTML. A partir de agora, sempre que citarmos a palavra container, estaremos nos referindo a uma caixa que servirá para armazenar conteúdo.

Você percebeu que a criação do layout foi feita basicamente com a tag <div>, porém, o HTML5 possui containers específicos para cada parte das nossas páginas, logo abordaremos esse tema utilizando os chamados containers semânticos.

Ids são fundamentais para os elementos HTML, por isso use-os e lembre-se de que devem ser únicos. Não use acentos para nomeá-los e dê preferência a letras minúsculas. Cuidado, nome de id é case sensitive, ou seja, deve referenciá-lo da mesma forma que criou.

Sempre que possível, comente seu código, ficará muito mais fácil para ler e entender aquilo que está criando. Código organizado e comentado é muito importante e ajuda muito na sua manutenção.

Wireframes são essenciais e ilustram como sua página ficará, por isso, antes de começar a codificação, faça o seu protótipo. Não tenha dúvidas de que isso o ajudará muito.

A figura “Parte da página Gog.com para o game Cuphead” apresenta uma pequena parte de uma das páginas do Gog.com (<https://www.gog.com>), uma grande plataforma de distribuição digital dos jogos. Pesquisamos pelo maravilhoso game CupHead e fomos redirecionados a uma página que tinha informações sobre o game e a possibilidade de efetuar sua compra. Veja a imagem e perceba que os containers estão bem visíveis, inclusive com os seus respectivos elementos, margin, padding, border e content.

É fácil perceber que o menu posicionado na parte superior da página é um container, logo abaixo temos outros quatro containers, dois para a galeria de imagens e outros dois que permitem adicionar o game ao carrinho de compras. Perceba também o alinhamento e o espaçamento entre eles, tudo isso conseguido por meio de formatações aplicadas aos elementos existentes no Box Model.

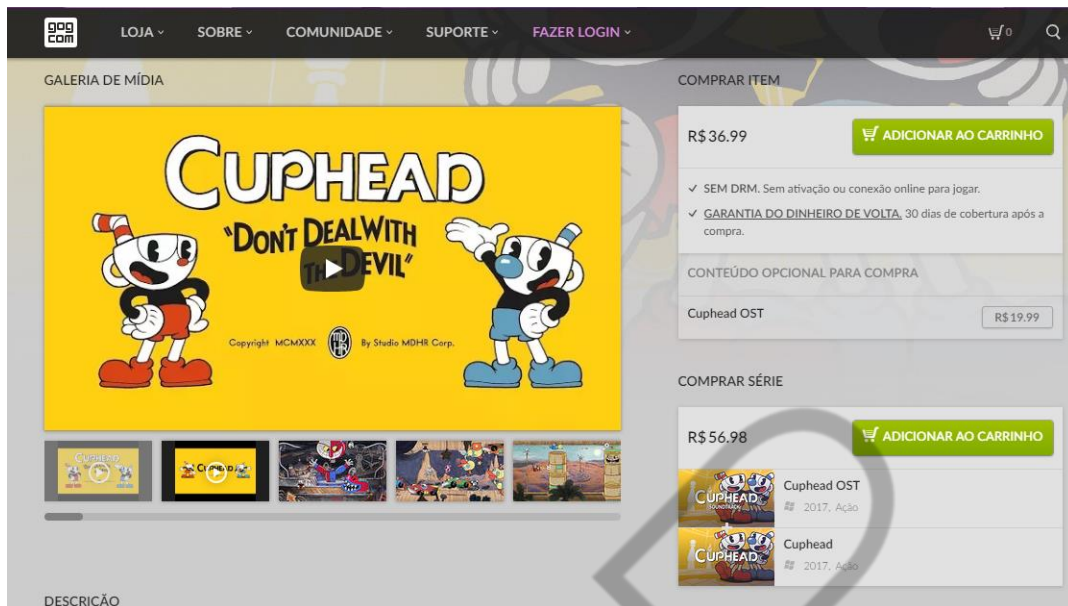


Figura 4.16 – Parte da página do Gog.com para o game Cuphead
Fonte: Gog (2018)

REFERÊNCIAS

ADOBE. **Photoshop**. [s.d.]. Disponível em: <<https://www.adobe.com/br/products/photoshop.html>>. Acesso em: 12 nov. 2020.

A PROPRIEDADE CSS border. **Maujor o dinossauro das CSS**. 2016. Disponível em: <<http://www.maujor.com/tutorial/propriedades-css-para-estilizacao-de-bordas.php>>. Acesso em: 12 nov. 2020.

BOX MODEL. [s.d.]. Disponível em: <<http://tableless.github.io/iniciantes/manual/css/box-model.html>>. Acesso em: 12 nov. 2020.

DUCKETT, J. **HTML&CSS projete e construa websites**. Rio de Janeiro: Alta Books, 2014.

MOQUPS. **Moqups Website**. [s.d.]. Disponível em: <<https://moqups.com/>>. Acesso em: 13 nov. 2020.

MOZILLA. **Box Model**. 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS/box_model>. Acesso em: 12 nov. 2020.

SILVA, M. S. **CSS3 – Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3**. São Paulo: Novatec, 2012

WIREFRAME. **Wireframe Website**. [s.d.]. Disponível em: <<https://wireframe.cc/>>. Acesso em: 12 nov. 2020.

W3SCHOOLS. **Box Model**. 2015. Disponível em: <https://www.w3schools.com/css/css_boxmodel.asp>. Acesso em: 12 nov. 2020.