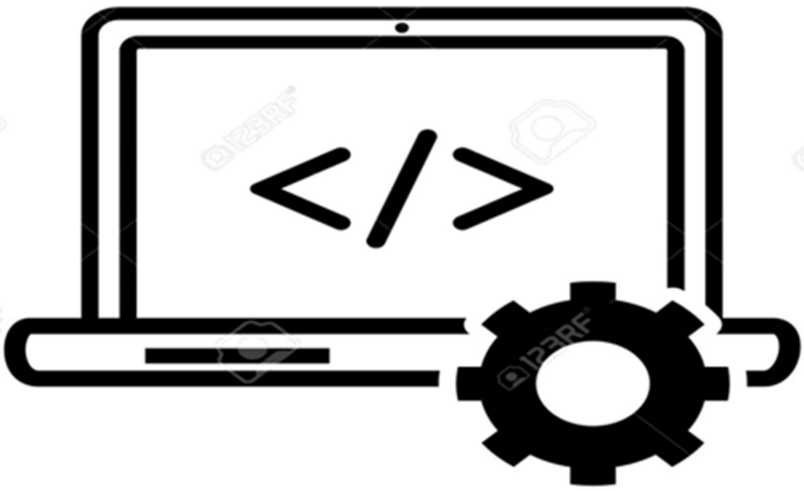
2019

Programação Web: HTML, CSS e Java Script



Geovanna Viana, Luís Henrique,
Luiz Felipe, Marcos Souza,
Michelly Mairink e Victor Hugo.
Projeto LabWeb – CEFET/RJ
24/08/2019

Sobre esta apostila

Esta apostila visa ensinar de uma maneira elegante, mostrando apenas o que é necessário e quando é necessário, no momento certo, poupando o leitor de assuntos que não costumam ser de seu interesse em determinadas fases do aprendizado.

Nós esperamos que você aproveite esse material. Todas as críticas e sugestões serão muito bem-vindas.

Esse material é parte integrante do treinamento Desenvolvimento web com HTML, CSS e JavaScript .

Projeto labweb.

Sumário

3 Introdução a JavaScript

CAPÍTULO 1

Introdução a HTML

1.1 SINTAXE DO HTML

O HTML é um conjunto de *tags* responsáveis pela marcação do conteúdo de uma página no navegador. No código que vimos antes, as tags são os elementos a mais que escrevemos usando a sintaxe <nomedatag>. Diversas tags são disponibilizadas pela linguagem HTML e cada uma possui uma funcionalidade específica.

No minicurso vimos, por exemplo, o uso da tag <h1>. Ela representa o título principal da página.

<h1>Mirror Fashion</h1>

Observe a sintaxe. Uma tag é definida com caracteres < e >, e seu nome (H1 no caso). Muitas tags possuem conteúdo, como o texto do título ("*Mirror Fashion*"). Nesse caso, para determinar onde o conteúdo acaba, usamos uma *tag de fechamento* com a barra antes do nome: </h1>.

Algumas tags podem receber **atributos** dentro de sua definição. São parâmetros usando a sintaxe de nome=valor. Para definir uma imagem, por exemplo, usamos a tag <img> e, para indicar qual imagem carregar, usamos o atributo src:

<img src= ".../imagens/logo_do_cefet.jpg" >.

Repare que a tag "img" não possui conteúdo por si só. Nesses casos, não é necessário usar uma tag de fechamento como antes no h1.

1.2 ESTRUTURA DE UM DOCUMENTO HTML

Um documento HTML válido precisa seguir obrigatoriamente a estrutura composta pelas tags <html>, <head> e <body> e a instrução <!DOCTYPE>. Vejamos cada uma delas:

A TAG <HTML>

Na estrutura do nosso documento, antes de tudo, inserimos uma tag <html>. Dentro dessa tag, é necessário declarar outras duas tags: <head> e <body>. Essas duas tags são "irmãs", pois estão no mesmo nível hierárquico em relação à sua tag "pai", que é <html>.

```
<html>
    <head></head>
```

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

A TAG <HEAD>

A tag <head> contém informações sobre nosso documento que são de interesse somente do navegador, e não dos visitantes do nosso site. São informações que não serão exibidas na área do documento no navegador.

A especificação obriga a presença da tag de conteúdo <title> dentro do nosso <head>, permitindo especificar o título do nosso documento, que normalmente será exibido na *barra de título* da janela do navegador ou na *aba* do documento.

Outra configuração muito utilizada, principalmente em documento cujo conteúdo é escrito em um idioma como o português, que tem caracteres como acentos e cedilha, é a configuração da codificação de caracteres, chamado de *encoding* ou *charset.*

Podemos configurar qual codificação queremos utilizar em nosso documento por meio da configuração de charset na tag <meta>. Um dos valores mais comuns usados hoje em dia é o **UTF-8**.

O **UTF-8** é a recomendação atual para encoding na web por ser amplamente suportada em navegadores e editores de código, além de ser compatível com praticamente todos os idiomas do mundo. É o que usaremos no minicurso.

```
<html>
    <head>
        <title>Mirror Fashion</title>
        <meta charset= “utf-8” >
    </head>
    <body>
     </body>
</html>
```

A TAG <BODY>

A tag <body> contém o corpo do nosso documento, que é exibido pelo navegador em sua janela. É necessário que o <body> tenha ao menos um elemento “filho”, ou seja, uma ou mais tags HTML dentro dele.

```
<html>
    <head>
        <title>Mirror Fashion</title>
        <meta charset= “utf-8” >
```

```
    </head>
     <body>
        <h1>A Mirror Fashion</h1>
     </body>
</html>
```

Nesse exemplo, usamos a tag <h1>, que indica um título.

A INSTRUÇÃO DOCTYPE

O DOCTYPE não é uma tag HTML, mas uma instrução especial. Ela indica para o navegador qual *versão do HTML* deve ser utilizada para renderizar a página.

Utilizaremos <!DOCTYPE html>, que indica para o navegador a utilização da versão mais recente do HTML, *a versão*.

1.3 TAGS HTML

O HTML é composto de diversas tags, cada uma com sua função e significado. O $HTML_5$, então, adicionou muitas novas tags, que veremos ao longo do minicurso.

Por enquanto, vamos focar em tags que representam *títulos, parágrafos e ênfase*.

TÍTULOS

Quando queremos indicar que um texto é um título em nossa página, utilizamos as tags de *heading* em sua marcação:

```
<h1>Mirror Fashion.</h1>
<h2>Bem-vindo à Mirror Fashion, sua loja de roupas e acessórios.</h2>
```

As tags de heading são tags de conteúdo e vão de <h1> a <h6>, seguindo a ordem de importância, sendo <h1> o título principal, o mais importante, e <h6> o título de menor importância.

Utilizamos, por exemplo, a tag <h1> para o nome, título principal da página, e a tag <h2> como subtítulo ou como título de seções dentro do documento.

A ordem de importância, além de influenciar no tamanho padrão de exibição do texto, tem impacto nas ferramentas que processam HTML. As ferramentas de indexação de conteúdo para buscas, como o Google, Bing ou Yahoo, levam! Levam em considerações essa ordem e relevância. Os navegadores especiais para acessibilidade também interpretam o conteúdo dessas tags de maneira a diferenciar seu conteúdo e facilitar a navegação do usuário pelo documento.

PARÁGRAFOS

Quando exibimos qualquer texto em nossa página, é recomendado que ele seja sempre conteúdo de alguma tag filha da tag <body>. A marcação mais indicada para textos comuns é a tag de *parágrafo*.

```
<p>Nenhum item na sacola de compras.</p>
```

Se houver vários parágrafos de texto, use várias dessas tags <p> para separá-los:

```
<p>Um parágrafo.</p>
<p>Outro parágrafo.</p>
```

MARCAÇÕES DE ÊNFASE

Quando queremos dar uma ênfase diferente a um trecho de texto, podemos utilizar as marcações de ênfase. Podemos deixar um texto "mais forte" com a tag ou deixar o texto com uma "ênfase acentuada" com a tag . Também há a tag <small>, que diminui o tamanho do texto.

Por padrão, os navegadores renderizarão o texto dentro da tag <Strong> em negrito e o texto dentro da tag em itálico. Existem ainda as tags <b> e <i>, que atingem o mesmo resultado visualmente, mas as tags e são mais indicadas por definirem nossa intenção de significado ao conteúdo, mais do que uma simples indicação visual.

```
<p>Compre suas roupas e acessórios na <Strong>Mirror Fashion</Strong>.</p>
```

1.4 IMAGENS

A tag <img> define uma imagem em uma página HTML e necessita de dois atributos preenchidos: src e alt. O primeiro aponta para o local da imagem e o segundo, um texto alternativo para a imagem caso essa não possa ser carregada ou visualizada.

O HTML$_5$ introduziu duas novas tags específicas para imagem: <figure> e <figcaption>, A tag <figure> define uma imagem com a conhecida tag <img>. Além disso, permite adicionar uma legenda para a imagem por meio da tag <figcaption>.

```
<figure>
   <img src= "img/produto1.png" alt= "Foto do produto" >
    <figcaption>calça por R$ 69,90</figcaption>
</figure>
```

1.5 A ESTRUTURA DOS ARQUIVOS DE UMA PÁGINA

Como um site é um conjunto de páginas web sobre um assunto, produto ou qualquer outra coisa, é comum todos os arquivos de um site estarem dentro de uma só pasta e, assim como um livro, é recomendado que exista uma "capa", uma página inicial que possa indicar para o visitante quais são as outras páginas que fazem parte desse projeto e como ele pode acessá-las, como se fosse o *índice* do site.

Esse índice é convenção adotada pelos servidores de páginas web. Se deseja que uma determinada pasta seja servida como um site e dentro dessa pasta exista um arquivo chamado *index.html*, esse arquivo será a página inicial a menos que alguma configuração determine outra página para esse fim.

Dentro da pasta do site, é recomendado que sejam criadas mais algumas pastas para manter separados os arquivos de imagens, as folhas de estilo CSS e os scripts.

1.6 EDITORES

Existem editores de texto como *Gedit* (www.gnome.org), *Sublime* (http://www.sublimetext.com/), Notepad++ (http://notepad-plus-plus.org), que possuem realce de sintaxe e outras ferramentas para facilitar o desenvolvimento de páginas.

1.7 EXERCÍCIOS: PRIMEIROS PASSOS COM HTML

1) Vamos importar todas as imagens dentro do projeto que criamos antes para que possamos usá-las nas páginas.

- Copie a pasta *mirror-fashion* de dentro da pasta *Caelum/43* para a área de trabalho de sua máquina.

- Verifique a pasta *img*, agora cheia de arquivos do design do site. Além desta pasta, teremos as pastas *js* e *css*, que ainda não usaremos.

2) Dentro da pasta *mirror-fashion*, vamos criar o arquivo *sobre.html* com a estrutura básica contendo o DOCTYPE e as tags html, head e body:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset= “utf-8” >
        <title>Sobre a Mirror Fashion</title>
     </head>
     <body>
     </body>
</html>
```

3) A página deve ter uma imagem com o logo da empresa, um título e um texto falando sobre ela.
Após copiar o texto para a página, coloque cada um dos parágrafos dentro de uma tag <p>. Coloque também o título *História* dentro de uma tag <h2>.
Use a tag <img> para o logo e a tag <h1> para o título. Seu HTML deve ficar assim, no final:

```
<img src= “img/logo.png” >

<h1>A Mirror Fashion</h1>

<p>
    A Mirror Fashion é a maior empresa...




</p>
```

4) Use negritos e itálicos para destacar partes importantes.
Use a tag <Strong> para a ênfase mais forte em negrito, por exemplo, para destacar o nome da empresa no texto do primeiro parágrafo:

```
<p> A <strong>Mirror Fashion </Strong> é a maior empresa comércio eletrônico...</p>.
```

Use também a ênfase com que deixará o texto em itálico. Na parte da *História*, coloque os nomes das pessoas e da família em .

5) A página deve ter duas imagens. A primeira apresenta o centro da *Mirror Fashion* e deve ser inserido *após o segundo parágrafo do texto*. A outra é uma imagem da *Família Pelho* e deve ser colocada após *o subtítulo da História*.

As imagens podem ser carregadas com a tag <img>, apontando seu caminho. Além disso, no $HTML_5$, podemos usar as tags <figure> e <figcaption> para destacar a imagem e colocar uma legenda em cada uma.

```
<figure>
   <img src= "img/centro-distribuicao.png" >
   <figcaption>Centro de distribuição da Mirror Fashion</figcaption>
</figure>
```

A imagem da família é a img/família-pelho.jpg e deve ser colocada na parte de *História*:

```
<figure>
    <img src= "img/familia-pelho.jpg" >
    <figcaption>Família Pelho</figcaptiom>
</figure>
```

Verifique o resultado no navegador. Devemos já ver o conteúdo e as imagens na sequência, mas sem um design muito interessante.

Dica: Uma prática recomendada, ligada à limpeza e utilizada para facilitar a leitura do código, é o uso correto de recuos, ou endentação, no HTML. Costumamos alinhar elementos "irmãos" na mesma margem e adicionar alguns espaços ou um tab para elementos "filhos".

CAPÍTULO 2

Introdução a CSS

2.1 Estilizando com CSS

Quando escrevemos o HTML, marcamos o conteúdo da página com tags que melhor representam o significado daquele conteúdo. Aí quando abrimos a página no navegador é possível perceber que o navegador mostra as informações com estilos diferentes.

Um h1, por exemplo, fica em negrito numa fonte maior. Parágrafos de texto são espaçados entre si, e assim por diante. Isso quer dizer que o navegador tem um *estilo padrão* para as tags que usamos. Mas claro, pra fazer sites bonitos vamos querer *customizar o design dos elementos* da página.

Antigamente, isso era feito no próprio HTML. Se quisesse um título verde, era só fazer. Mas agora, surgiu o **CSS**, que é outra linguagem, separada do HTML, com o objetivo único de cuidar da estilização da página. A vantagem é que o CSS é bem mais robusto que o HTML para estilização, como veremos. Mas, escrever formatação visual misturado com conteúdo de texto no HTML se mostrou algo bem impraticável. O CSS resolve isso separando as coisas; regras de estilo não aparecem mais no HTML, apenas no CSS.

2.2 Sintaxe e inclusão de CSS

A sintaxe do CSS tem estrutura simples: é uma declaração de propriedades e valores separados por um sinal de dois pontos ":", e cada propriedade é separada por um sinal de ponto e vírgula ";" da seguinte maneira:

```
Background-color: yellow;
Color: blue;
```

O elemento que receber essas propriedades será exibido com o texto na cor azul e com o fundo amarelo. Essas propriedades podem ser declaradas de três maneiras diferentes.

ATRIBUTO STYLE

A primeira delas é como um atributo style e no próprio elemento:

```
<p style= "color: blue; background-color: yellow;" >
O conteúdo desta tag será exibido em azul com fundo amarelo no navegador!
</p>
```

Mas tínhamos acabado de discutir que uma das grandes vantagens do CSS era manter as regras de estilo fora do HTML. Usando esse atributo *style* não parece que fizemos isso. Justamente por isso não se recomenda esse tipo de uso na prática, mas sim os que veremos a seguir.

A TAG STYLE

A outra maneira de se utilizar o CSS é declarando suas propriedades dentro de uma tag <style>.

Como estamos declarando as propriedades visuais de um elemento em outro lugar do nosso documento, precisamos indicar de alguma maneira a qual elemento nos referimos. Fazemos isso utilizando um **seletor CSS**. É basicamente uma forma de buscar certos elementos dentro da página que receberão as regras visuais que queremos.

No exemplo a seguir, usaremos o seletor que pega todas as tags *p* e altera sua cor e background:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset= "utf-8" >
      <title>Sobre a Mirror Fashion</title>
      <style>
        P{
          Background-color: yellow;
          Color: blue;
         }
      </style>
    </head>
     <body>
        <p>
           O conteúdo desta tag será exibido em azul com fundo amarelo!
        </p>
        <p>
           <Strong>Também</Strong> será exibido em azul com fundo amarelo!
         </p>
       </body>
</html>
```

O código anterior da tag <style> indica que estamos alterando a cor e o fundo de todos os elementos com tag p. Dizemos que selecionamos esses elementos pelo nome de sua tag, e aplicamos certas propriedades CSS apenas neles.

ARQUIVO EXTERNO

A terceira maneira de declararmos os estilos do nosso documento é com um arquivo externo, geralmente com a extensão *.css*. Para que seja possível declarar nosso CSS em um arquivo à parte, precisamos indicar em nosso documento HTML uma ligação entre ele e a folha de estilo.

Além da melhor organização da página, a folha de estilo traz ainda as vantagens de manter nosso HTML mais limpo e do reaproveitamento de uma mesma folha de estilos para diversos documentos.

A indicação de uso de uma folha de estilos externa deve ser feita dentro da tag <head> do nosso documento HTML:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset= “utf-8” >
        <title>Sobre a Mirror Fashion</title>
        <link rel= “stylesheet” href= “estilos.css” >
     </head>
     <body>
         <p>
            O conteúdo desta tag será exibido em azul com fundo amarelo!
         </p>
         <p>
            <Strong>Também</Strong> será exibido em azul com fundo amarelo!
         </p>
      </body>
</html>
```

E dentro do arquivo *estilos.css* colocamos apenas o conteúdo do CSS:

```
P{
  color: blue;
  Backgroun-color: yellow;
}
```

2.3 Propriedades tipográficas e fontes

Da mesma forma que alteramos cores, podemos alterar o texto. Podemos definir fontes com o uso da propriedade font-family.

A propriedade font-family pode receber seu valor com ou sem aspas. No primeiro caso, passaremos o nome do arquivo de fonte a ser utilizado, no último, passaremos a família da fonte.

Por padrão, os navegadores mais conhecidos exibem texto em um tipo que conhecemos como “serif”. As fontes mais conhecidas (e comumente utilizadas como padrão) são “times”

e “Times New Roman”, dependendo do sistema operacional. Elas são chamadas de **fontes serifadas** pelos pequenos ornamentos em suas terminações.

Podemos alterar a *font-family* que queremos utilizar em nossa página para a “sans-serif” (sem serifas), que contém, por exemplo, as fontes “Arial” e “Helvetica”. Podemos também declarar que queremos utilizar uma família de fontes “monospace” como, por exemplo, a fonte “Courier”.

```
h1{
    font-family: serif;
}
h2{
  font-family: sans-serif;
}
P{
  Font-family: monospace;
}
```

É possível, declararmos o nome de algumas fontes que gostaríamos de verificar se existem no computador, permitindo que tenhamos um controle melhor da forma como nosso texto será exibido. Normalmente, declaramos as fontes mais comuns, e existe um grupo de fontes que são consideradas “seguras” por serem bem populares.

2.4 Alinhamento e decoração de texto

Já vimos uma série de propriedades e subpropriedades que determinam o estilo do tipo (fonte). Vamos conhecer algumas maneiras de alterarmos as disposições dos textos.

Uma das propriedades mais simples, porém muito utilizada, é a que diz respeito ao alinhamento de texto: a propriedade text-align.

```
P{
  Text-align: right;
}
```

O exemplo anterior determina que todos os parágrafos da nossa página tenham o texto alinhado para a direita. Também é possível determinar que um elemento tenha seu conteúdo alinhado ao centro ao definirmos o valor *center* para a propriedade *text-align*, ou então definir que o texto deve ocupar toda a largura do elemento aumentando o espaçamento entre as palavras com o valor *justify.* O padrão é que o texto seja alinhado à esquerda, com o valor *left*, porém é importante lembrar que essa propriedade propaga-se em cascata.

É possível configurar também uma série de espaçamentos de texto com o CSS:

```
P{
  line-height: 3px; /* tamanho da altura de cada linha */
  letter-spacing: 3px; /* tamanho do espaço entre cada letra */
  word-spacing: 5px; /* tamanho do espaço entre cada palavra */
  text-indent: 30px; /* tamanho da margem da primeira linha do texto */
}
```

2.5 Imagem de fundo

A propriedade background-image permite indicar um arquivo de imagem para ser exibido ao fundo do elemento. Por exemplo:

```
h1{
  background-image: url(sobre-background.jpg);
}
```

Com essa declaração, o navegador vai requisitar um arquivo *sobre-background.jpg*, que deve estar na mesma pasta do arquivo CSS onde consta essa declaração.

2.6 Bordas

As propriedades do CSS para definirmos as bordas de um elemento nos apresentam uma série de opções. Podemos, para cada borda de um elemento, determinar sua cor, seu estilo de exibição e sua largura. Por exemplo:

```
body {
    border-color: red;
    border-style: solid;
    border-width: 1px;
}
```

Para que o efeito da cor sobre a borda surta efeito, é necessário que a propriedade *border-style* tenha qualquer valor diferente do padrão *none*.

2.7 Exercícios: Primeiros passos com CSS

1) Aplicaremos um pouco de estilo em nossa página usando CSS. Dentro da pasta *css*, crie um arquivo *sobre.css*, que conterá nosso código de estilo para essa página.

Em primeiro lugar, precisamos carregar o arquivo **sobre.css** dentro da página **sobre.html** com a tag <link> dentro da tag <head>:

<link rel= “stylesheet” href= “css/sobre.css”>

Para o elemento <body>, altere a sua cor e sua fonte base por meio das propriedades *color* e *font-family*:

```
body{
    color: #333333;
    font-family: “Lucida Sans Unicode”, “Lucida Grande”, sans-serif;
}
```

O título principal deve ter um fundo estampado com a imagem img/sobre-background.jpg. Use a propriedade background-image pra isso. Aproveite e coloque uma borda sutil nos subtítulos, para ajudar a separar o conteúdo.

```
h1{
    background-image: url (.../img/sobre-background.jpg);
}
h2{
    border-bottom: 2px solid #333333;
}
```

Acerte também a renderização das figuras. Coloque um fundo cinza, uma borda sutil, deixe a legenda em itálico com *font-style* e alinhe a imagem e a legenda no centro com *text-align*.

```
figure{
    background-color: #F2EDED;
    border: 1px solid #ccc;
     text-align: center;
}
Figcaption{
    Font-style: italic;
}
```

Teste o resultado no navegador. Nossa página começa a pegar o estilo da página final!

2.8 Cores na web

Propriedades como background-color, color, border-color, entre outras aceitam uma cor como valor. Existem várias maneiras de definir cores quando utilizamos o CSS.

A primeira, mais simples, é usando o nome da cor:

```
h1{
   color: red;
}
h2{
   background: yellow;
}
```

O difícil é acertar a exata variação de cor que queremos no design. Por isso, é bem incomum usarmos cores com seus nomes. O mais comum é definir a cor com base em sua composição RGB.

RGB é um sistema de cor bastante comum aos designers. Ele permite especificar ate 16 milhões de cores como uma combinação de três cores base: Vermelho (Red), Verde (Green), Azul (Blue). Podemos escolher a intensidade de cada um desses três canais básicos, numa escala de *0 a 255*.

Um amarelo forte, por exemplo, tem 255 de Red, 255 de Green e o de Blue *(255, 255, 0)*. Se quiser um laranja, basta diminuir um pouco o verde *(255, 200, 0)*. E assim por diante.

```
h3{
   color: rgb (255, 200, 0);
}
```

Essa sintaxe funciona nos browsers mais modernos, mas não é a mais comum na prática, por questões de compatibilidade. O mais comum é a **notação hexadecimal**, que acabamos usando no exercício anterior ao escrever **#F$_2$EDED**. Essa sintaxe tem suporte universal nos navegadores e é mais curta de escrever, apesar de ser mais enigmática.

```
h3{
  background: #F2EDED;
}
```

No fundo, porém, é a mesma coisa. Na notação hexadecimal (que começa com #), temos 6 caracteres. Os primeiros dois indicam o canal Red, os dois seguintes, o Green, e os dois

últimos, Blue. Ou seja, RGB. E usamos a matemática pra escrever menos, trocando a base numérica de decimal para hexadecimal.

Na base hexadecimal, os algarismos vão de zero a quinze (ao invés do zero a nove da base decimal comum). Para representar os algarismos de dez a quinze, usamos letras de A a F. Nessa sintaxe, portanto, podemos utilizar números de 0-9 e letras de A-F.

Há uma conta por trás dessas conversões, mas seu editor de imagens deve ser capaz de fornecer ambos os valores para você sem problemas. Um valor 255 vira FF na notação hexadecimal. A cor #F_2EDED, por exemplo, é equivalente a rgb (242, 237, 237), um cinza claro.

Vale uma dica quanto ao uso de cores hexadecimais, toda vez que os caracteres presentes na composição da base se repetirem, estes podem ser simplificados. Então um número em hexadecimal 3366FF, pode ser simplificado para 36f.

2.9 Espaçamento e margem

Utilizamos a propriedade padding para espaçamento e margin para margem. Vejamoa cada uma e como elas diferem entre si.

PADDING

A propriedade padding é utilizada para definir uma margem interna em alguns elementos (por margem interna queremos dizer a distância entre o limite do elemento, sua borda, e seu respectivo conteúdo) e tem as subpropriedades listadas a seguir:

- Padding-top
- Padding-right
- Padding-bottom
- Padding-left

Essas propriedades aplicam uma distância entre o limite do elemento e seu conteúdo acima, à direita, abaixo e à esquerda respectivamente. Essa ordem é importante para entendermos como funciona a *shorthand property* do padding.

Podemos definir todos os valores para as subpropriedades do padding em uma única propriedade, chamada exatamente de padding, e seu comportamento é descrito nos exemplos a seguir:

Se passado somente um valor para a propriedade padding, esse mesmo valor é aplicado em todas as direções.

```
p{
  padding: 10px;
}
```

Se passado dois valores, o primeiro será aplicado acima e abaixo (equivalente a passar o mesmo valor para padding-top e padding-bottom) e o segundo será aplicado à direita e à esquerda (equivalente ao mesmo valor para padding-right e padding-left).

```
p{
  padding: 10px 15px;
}
```

Se passados três valores, o primeiro será aplicado acima (equivalente a pdding-top), o segundo será aplicado à direita e à esquerda (equivalente a passar o mesmo valor para padding-right e padding-left_ e o terceiro valor será aplicado abaixo do elemento (equivalente a padding-bottom).

```
p{
  padding: 10px 20px 15px;
}
```

Se passados quatro valores, serão aplicados respectivamente a padding-top, padding-right, padding-bottom e padding-left. Para facilitar a memorização dessa ordem, basta lembrar que os valores são aplicados em sentido horário.

```
p{
  Padding: 10px 20px 15px 5px;
}
```

MARGIN

A propriedade margin é bem parecida com a propriedade padding, exceto que ela adiciona espaço após o limite do elemento, ou seja, é um espaçamento além do elemento em si. Além das subpropriedades listadas a seguir, há a *shorthand property* margin que se comporta da mesma maneira que a *shorthand property* do padding vista no tópico anterior.

- margin-top
- margin-right
- margin-bottom
- margin-left

Há ainda uma maneira de permitir que o navegador defina qual será a dimensão da propriedade padding ou margin conforme o espaço disponível na tela: definimos o valor auto para a margem ou o espaçamento que quisermos.

No exemplo a sequir, definimos que um elemento não tem nenhuma margem acima ou abaixo de seu conteúdo e que o navegador define uma margem igual para ambos os lados de acordo com o espaço disponível:

```
p{
  margin: 0 auto;
}
```

DIMENSÕES

É possível determinar as dimensões de um elemento, por exemplo:

```
p{
  background-color: red;
  height: 300px;
  width: 300px;
}
```

Todos os parágrafos do nosso HTML ocuparão 300 pixels de largura e de altura, com cor de fundo vermelha.

2.10 Exercícios: Lista e margens

1) Ainda na página **sobre.html**, crie um subtítulo chamado **Diferenciais** e, logo em seguida, uma lista de diferenciais. Use <h2> para o subtítulo, <ul> para a lista e <li> para os itens da lista.

Dica: você pode copiar o texto dos diferenciais do arquivo **diferenciais.txt.**

```
<h2>Diferenciais</h2>

<ul>
   <li>Menor preço do varejo, garantido</li>
   <li>Se você achar uma loja mais barata, leva o produto de graça</li>]
   . . . .
</ul>
```

Teste o resultado no navegador.

2) Podemos melhorar a apresentação da página acertando alguns espaçamentos usando várias propriedades do CSS, como *margin, padding e text-indent.*

```
h1{
    padding: 10px;
}
h2{
    margin-top: 30px;
}
p{
   padding: 0 45px;
   text-indent: 15px;
}
figure{
      padding: 15px;
      margin: 30px;
}
figcaption{
       margin-top: 10px;
}
```

Veja o resultado no navegador.

3) Para centralizar o body como no design, podemos usar o truque de colocar um tamanho fixo e margens auto na esquerda e na direita:

```
body{
      margin-left: auto;
      margin-right: auto;
      width: 940px;
}
```

Verifique mais uma vez o resultado.

2.11 Links HTML

Quando precisamos indicar que um trecho de texto se refere a um outro conteúdo, seja ele no mesmo documento ou em outro endereço, utilizamos a tag de âncora <a>.

Existem dois diferentes usos para as âncoras. Um deles é a definição de links:

```
<p>
    Visite o site da <a href= http://www.caelum.com.br>Caelum </a>
```

</p>

Note que a âncora está demarcando apenas a palavra “Caelum” de todo o conteúdo do parágrafo exemplificado. Isso significa que, ao clicarmos com o cursor do mouse na palavra “Caelum”, o navegador redirecionará o usuário para o site da Caelum, indicado no atributo href.

Outro uso para a tag de âncora é a demarcação de destinos para links dentro do próprio documento, o que chamamos de bookmark.

```
<p>Mais informações <a href= “#info” >aqui</a>.</p>
<p>Conteúdo da página . . .</p>

<h2 id= “info”>Mais informações sobre o assunto:</h2>
<p>Informações . . .</p>
```

De acordo com o exemplo acima, ao clicarmos sobre a palavra “aqui”, demarcada com um link, o usuário será levado à porção da página onde o bookmark “info” é visível. Bookmark é o elemento que tem o atributo id.

É possível, com o uso de um link, levar o usuário a um bookmark presente em outra página.

```
<a href= “http://www.caelum.com.br/curso/wd43/#contato” >
   Entre em contato sobre o curso
</a>
```

O exemplo acima fará com que o usuário que clicar no link seja levado à porção da página indicada no endereço, especificamente no ponto onde o bookmark “contato” seja visível.

2.12 Exercícios: Links

1) No terceiro parágrafo do texto, citamos o centro de distribuição na cidade de *jacarezinho, no Paraná.* Transforme esse texto em um link externo apontando para o mapa no Google Maps.

Use a tag <a> para criar link para o Google Maps:

```
<a href= “https://maps.google.com.br/?q=Jacarezinho”>
   Jacarezinho, no Paraná
</a>
```

Teste a página no navegador e acesse o link.

2) Durante o minicurso, vamos criar várias páginas para o site da Mirror Fashion, como uma página inicial (chamada **index.html**).

Queremos, nessa página de *Sobre* que estamos fazendo, linkar para essa página. Por isso, vamos criá-la agora na pasta *mirror-fashion* com a estrutura básica e um parágrafo indicando em qual página o usuário está. Não se preocupe, ela será incrementada em breve.

Crie a página *index.html*:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Mirror Fashion</title>
        <meta charset= “utf-8” >
      </head>
      <body>
          <h1>Olá, sou o index.html!</h1>
      </body>
</html>
```

Adicione um link interno na nossa *sobre.html* apontando para esta página que acabamos de criar. O terceiro parágrafo do texto possui referência a esta página mas ainda sem link. Crie link lá:

```
. . . Acesse <a href= “index.html” >nossa loja </a>. . .
```

Repare como apenas envolvemos o texto a ser linkado usando a tag <a>.

Veja o resultado.

3) Se reparar bem, ainda nesse terceiro parágrafo de texto, há referências textuais para as outras seções da nossa página, em particular para a *História e os Diferenciais*. Para facilitar a navegação do usuário, podemos transformar essas referências em âncoras para as respectivas seções no HTML.

Para isso, adicione um *id* em cada um dos subtítulos para identifica-los:

```
<h2 id= “historia” >História</h2>
. . .
<h2 id= “diferenciais” >Diferenciais</h2>
```

Agora que temos os *ids* dos subtítulos preenchidos, podemos criar uma âncora para eles:

```
. . . Conheça também nossa <a href= “#historia” >história</a> e
```

Nossos diferenciais. . . .

Veja o resultado em seu navegador.

2.13 Fluxo do documento e float

Suponhamos que, por uma questão de design, a imagem da família Pelho deva vir ao lado do parágrafo.

Isso não acontece por padrão. Repare que, observando as tags HTML que usamos até agora, os elementos da página são desenhados um em cima do outro. É como se cada elemento fosse uma caixa e o padrão é empilhar essas caixas verticalmente. Mais pra frente vamos entender melhor esse algoritmo, mas agora o importante é que ele atrapalha esse nosso design.

Temos um problema: a tag <figure> ocupa toda a largura da página e aparece empilhada no fluxo do documento, não permitindo que outros elementos sejam adicionados ao seu lado.

Este problema pode ser solucionado por meio da propriedade *float*. Esta propriedade permite que tiremos um certo elemento do fluxo vertical do documento o que faz com que o conteúdo abaixo dele flua ao seu redor. Na prática, vai fazer exatamente o layout que queremos.

Em nosso exemplo, o conteúdo do parágrafo tentará fluir ao redor da nossa imagem com *float.* Perceba que houve uma perturbação do fluxo HTML, pois agora a nossa imagem parece existir fora do fluxo.

2.14 Exercícios: seletores CSS e flutuação de elementos

1) Temos uma <figure> com a imagem do centro de distribuição que queremos centralizar na página (*margin auto*) e acertar o tamanho (*width*).

Para aplicar essas regras apenas a esse figure e não a todos da página, vamos usar o ID. Coloque um id na <figure> para podermos estilizá-la especificamente via CSS:

<figure id= "centro- distribuicao" >

. . . .

Adicionando o CSS:

```
#centro-distribuicao {
    margin-left: auto;
```

```
    margin-right: auto;
    width: 550px;
}
```

Teste no navegador. Compare o resultado com a outra figura que não recebeu o mesmo estilo.

2) Crie um rodapé para a página utilizando uma , que deve ser inserida como último elemento dentro da tag <body>:

```
<div id= "rodape" >
    <img src= "img/logo.png" >

    &copy; copyright Mirror Fashion
</div>
```

Teste o resultado.

3) Assim como fizemos para os títulos e subtítulos, estilize o nosso rodapé. Repare no uso do id via CSS para selecionarmos apenas o elemento específico que será estilizado. Repare também no uso do *seletor de descendentes* para indicar um elemento que está dentro de outro.

```
#rodape {
    color: #777;
    margin: 30px 0;
    padding: 30px 0;
}

#rodape img {
    margin-right: 30px;
    vertical-align: middle;
    width: 94px;
}
```

Teste o resultado final no navegador.

4) Queremos que a imagem da *Família Pelho* esteja flutuando a direita no texto na seção sobre a *História* da empresa. Para isso, use a propriedade *float* no CSS.

Como a <figure> com a imagem da família Pelho ainda não possui id, adicione um:
<figure id= "família-pelho" >

. . . .

Agora podemos referenciar o elemento através de seu id em nosso CSS, indicando a flutuação e uma margem para espaçamento:

```
#familia-pelho {
      float: right;
      margin: 0 0 10px 10px;
}
```

Teste o resultado.

CAPÍTULO 3

Introdução a JavaScript

3.1 Características da linguagem

O JavaScript, como o próprio nome sugere, é uma linguagem de *scripting*. Uma linguagem de *scripting* é comumente definida como uma linguagem de programação que permite ao programador controlar uma ou mais aplicações de terceiros. No caso do JavaScript, podemos controlar alguns comportamentos dos navegadores através de trechos de código que são enviados na página HTML.

Outra característica comum nas linguagens de *scripting* é que normalmente elas são linguagens interpretadas, ou seja, não dependem de compilação para serem executadas. Essa característica é presente no JavaScript: o código é interpretado e executado conforme é lido pelo navegador, linha a linha, assim como o HTML.

O JavaScript também possui grande tolerância a erros, uma vez que conversões automáticas são realizadas durante operações. Como será visto no decorrer das explicações, nem sempre essas conversões resultam em algo esperado, o que pode ser fonte de muitos bugs, caso não conheçamos bem esse mecanismo.

O Script do programador é enviado com o HTML para o navegador, mas como o navegador saberá diferenciar o script de um código html? Para que essa diferenciação seja possível, é necessário envolver o script dentro da tag <script>.

3.2 A tag script

Para rodar JavaScript em uma página Web, precisamos ter em mente que a execução do código é instantânea. Para inserirmos um código JavaScript em uma página, é necessário utilizar a tag <script>:

```
<script>
    alert ( “Olá, Mundo!” );
</script>
```

O exemplo acima é um “hello world” em JavaScript utilizando uma função do navegador, a função **alert**.

A tag <script> pode ser declarada dentro da tag <head> assim como na tag <body>, mas devemos ficar atentos, porque o código é lido imediatamente dentro do navegador. Veja a consequência disso nos dois exemplos abaixo:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset= “utf-8” >
      <title>Aula de JS</title>
      <script>
          alert ( “Olá, Mundo!” );
      </script>
    </head>
    <body>
       <h1>JavaScript</h1>
       <h2>Linguagem de programação</h2>
    </body>
</html>
```

Repare que, ao ser executado, o script trava o processamento da página. Imagine um script que demore um pouco mais para ser executado ou que exija alguma interação do usuário como uma confirmação. Não seria interessante carregar a página toda primeiro antes de sua execução por uma questão de performance e experiência para o usuário?

Para fazer isso, basta removermos o script do head, colocando-o no final do body:

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

```
        <meta charset= “utf-8” >
        <title>Aula de JS</title>
    </head>
    <body>
        <h1>JavaScript</h1>
        <h2>Linguagem de programação</h2>
        <script>
            Alert (“Olá, Mundo!”);
        </script>
    </body>
</html>
```

Devemos sempre colocar o script antes de fecharmos a tag </body>? Na maioria das vezes sim.

JAVASCRIPT EM ARQUIVO EXTERNO

Imagine ter que escrever o script toda vez que ele for necessário. Se ele for utilizado em outra página? Por isso é possível importar scripts dentro da página utilizando também a tag <script>:

No arquivo HTML

```
<script src= “js/hello.js”></script>
```

Arquivo externo js/hello.js

```
alert ( “Olá, Mundo!” );
```

Com a separação do script em arquivo externo é possível reaproveitar alguma funcionalidade em mais de uma página.

3.3 Sintaxe básica

Operadores

Teste algumas contas digitando diretamente no console:

```
> 12 + 13
25
> 14 * 3 42
> 10 - 4
6
> 25 / 5
5
> 23 % 2
```

Variáveis

Para armazenarmos um valor para uso posterior, podemos criar uma variável:

> var resultado = 102 / 17;

Undefined

No exemplo acima, guardamos o resultado de 102 / 17 na variável resultado. O resultado de criar uma variável é sempre undefined. Para obter o valor que guardamos nela ou mudar o seu valor, podemos fazer o seguinte:

> resultado
6
> resultado = resultado + 10
16
> resultado
16

Também podemos alterar o valor de uma variável usando as operações básicas com uma sintaxe bem compacta:

> var idade = 10; // undefined

> idade += 10; // idade vale 20

> idade -= 5; // idade vale 15

> idade /= 3; // idade vale 5

*> idade *= 10; // idade vale 50*

Tipos de dados

Não são apenas números que podemos salvar numa variável. O Javascript tem vários tipos de dados.

Number

Com esse tipo de dados é possível executar todas as operações que vimos anteriormente:

var pi = 3.14159;

var raio = 20;

*var perimetro = 2 * pi * raio*

String

Uma string em JavaScript é utilizada para armazenar trechos de texto:

var escola = "CEFET/RJ";

Para exibirmos o valor da variável *escola* fora do console, podemos executar o seguinte comando:

```
alert(escola);
```

O comando alert serve para criação de popups com algum conteúdo de texto que colocarmos dentro dos parênteses. O que acontece com o seguinte código?

```
var numero = 30;
alert(numero)
```

O número 30 é exibido sem problemas dentro do popup. O que acontece é que qualquer variável pode ser usada no alert . O JavaScript não irá diferenciar o tipo de dados que está armazenado numa variável, e se necessário, tentará converter o dado para o tipo desejado.

3.5 Arrays

Neste capítulo faremos uma revisão do objeto Array, uma vez que já o estamos utilizando desde o início do livro. Percebemos que todos os objetos de um mesmo tipo, ao se unirem dentro de um mesmo documento, podem ser agrupados dentre de um array.

Matrizes serão o outro tópico de estudo dessa fase. Veremos que seu uso é bastante e simples e também aprenderemos que elas são tão somente arrays de outros arrays. Então, mãos a obra.

3.6 Objeto Array

O objeto Array permite que representemos uma coleção de objetos do mesmo tipo. Por exemplo, todas as imagens de uma página podem ser representadas pelo array images. Existem duas formas simples de se criar um array. A primeira é informando diretamente ao construtor, os elementos desejados. Observem:

```
... carros = new Array( "GOL", "CORSA", "PALIO" );
```

- Ao se instanciar um array lembrem−se de que a primeira letra da palavra array deve estar maiúscula.
- Arrays são criados através do comando new.

A outra forma de se criar o array é a seguinte:

```
... carros = new Array( 3 );//indica que o array terá três elementos
 ... //Neste caso os elementos são inseridos através de seus índices
... carros[0] = "GOL";
... carros[1] = "CORSA";
... carros[2] = "PALIO";
```

Uma vez criados, a forma mais fácil de se obter os valores novamente, é através de um laço de repetição. Vamos a um exemplo prático:

```
<html>
    <head>
        <tiltle>Trabalhando com arrays</title>
```

```
<script language= “Javascript”>
    notas= new Array (3);

  //Este laço obtém do usuário valores que serão atribuídos ao array de nota
  For ( i = 0; i < 3; i++ ) {
        Notas[ i ] = prompt ( ‘informe a nota do aluno’ + ( i + 1 ), ‘’
      }
 //Uma vez o array criado, iremos imprimir os valores na tela
 for ( i = 0; i < 3; i++ ) {
        document.write ( ‘Aluno ‘, i +1, ‘:’, notas [ i ],  ‘<br>’ );
      }
</script>
</head>
<body>
</body>
</html>
```

O acesso a cada elemento é feito pela sua posição dentro do array, como mostrado no exemplo. O laço de repetição é muito útil, porque sua estrutura permite o acesso de todos os elementos de array em uma só rotina de programação. Imaginem uma impressão de relatório de notas de 1000 alunos feitas sem o recurso do laço?

3.7 Exercícios de fixação

1- Declare as seguintes variáveis e atribua valores correspondentes aos seus tipos: nome(string), idade(inteiro), peso(double). 1. 2. Utilize uma estrutura de controle para descobrir se o valor de sua idade é par ou ímpar. 3. Faça um laço para imprimir todos os anos, do ano atual até o ano de seu nascimento.

Resposta:

Parte 1

```
var nome = "Luis";
 var idade = 18;
var peso = 80;
```

Parte 2

```
01. <script language="Javascript">
02. function parImpar( num ) {
03. if( num%2 == 0 ) //se resto da divisão por 2 for zero, o número é par
04. document.write( 'O número ', num, ' é par' );
05. else
06. document.write( 'O número ', num, ' é ímpar' );
07. }
08.
09. parImpar( 5 );
10. </script>
```

Parte 3

```
01. <script language="Javascript">
02. for( i = 2019; i >=2000; i-- )
03. document.write( i, '<br>' );
04. </script>
```

Referências:

http://www.lyfreitas.com.br/ant/pdf/javascript.pdf

https://www.caelum.com.br/download/caelum-html-css-javascript.pdf