Front-end do zero

Ajudando você a criar um castelo de código

Iuri Silva

Sobre o e-book

Esse é um pequeno guia de passos para iniciante começar no desenvolvimento front-end. Espero que você possa aprender muitas coisas ao ler cada página deste e-book.

Faça todos os testes você mesmo, faça exercícios, busque informações e se aprofunde.

Observação importante:

O desenvolvimento do e-book não foi concluído ainda, falta o fundamentos de JavaScript.

Sinta-se à vontade em ajudar a comprar meu café, assim continuo criando conteúdos de forma gratuita :)

Chave do pix: iuricold99@gmail.com

Grande abraço!
by @iuricode

Fundamentos HTML

O que é HTML

HTML, ou *Hypertext Markup Language* é uma linguagem de marcação (*não de programação*) da web - cada vez que você carrega uma página da web, você está carregando um código HTML. Pense em HTML como o esqueleto de uma página da web, ele é responsável pelos textos, links, listas e imagens - ele oferece conteúdos.

Iniciando

HTML é escrito em arquivos *.html*. Para criar uma página HTML é fácil, entre em seu editor de código (*se não tiver recomendo o visual studio code*) e salve o arquivo em branco como *pagina.html* (*você pode nomeá-lo como quiser, mas não esqueça do .html*).

Estrutura de uma página HTML

A estrutura inicial do seu html será essa:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

Vou explicar sobre elas!

<!DOCTYPE html> - Essa tag (*não tem fechamento dela*) informa ao seu navegador que o arquivo faz parte de um documento HTML5.

<html> - Representa a raiz de um documento HTML.

<head> - O head contém informações sobre sua página, mas não é o conteúdo que vai aparecer na sua página. Ela conterá coisas como: links para folhas de estilo (*CSS*), título da página, links de fontes e tudo aquilo que você quiser linkar.

<body> - O body contém todo o conteúdo que vai aparecer na sua página. Todo o código que você escrever estará dentro dele.

Sintaxe

Os elementos HTML são escritos usando tags. Todas as tags tem uma chave de abertura e fechamento *<tag>*. A tag de fechamento que tem uma barra após o primeiro colchete *</tag>*.

```
<tag>iuricode</tag>
```

Por exemplo, se você deseja criar um parágrafo, usaremos as chaves de abertura * * e fechamento * *:

```
<p>Programador sem café é um poeta sem poesia.</p>
```

Os elementos podem entrar dentro de outros elementos:

```
<pai>
  <filho>
    Esta tag está dentro de outra tag,
    também conhecida como a tag filho.
  </filho>
</pai>
```

Indentação adequada

As quebras de linha entre suas tags são super importantes para escrever um bom código HTML.

```
<! DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h1>Aqui temos um título</h1>
    <p>E aqui um parágrafo</p>
  </body>
</html>
```

Abaixo temos um indentação não recomendada:

```
<! DOCTYPE html> <html> <head> </head> <body> <h1>
Não faça isso! </h1> </body> </html>
```

Tags

Agora vamos apresentar as principais tags do HTML.

Título

As tags de títulos são representadas em até seis níveis. *<h1>* é o nível mais alto e *<h6>* é o mais baixo. Quanto maior for o nível da tag, maior vai ser o tamanho da fonte e a importância.

```
<h1>Título do h1</h1>
<h2>Título do h2</h2>
<h3>Título do h3</h3>
<h4>Título do h4</h4>
<h5>Título do h5</h5>
<h6>Título do h6</h6>
```

Texto

As tags de texto, definem diferentes formatações para diversos tipos de texto. Desde estilos de fonte, parágrafos, quebra de linha ou até mesmo spans. Enfim iremos conhecê-las:

** ** - Sendo a principal tag de texto, é usada para constituir um parágrafo.

```
<p>Este é o primeiro parágrafo do texto.</p>
```

**** - Mesmo tendo a sua funcionalidade parecida com o uso dos parágrafos. Spans geralmente são utilizados para guardar uma pequena informação.

```
<span>
  <h1>Sou um título</h1>
  <p>Este é o primeiro parágrafo</p>
  <p>Este é o segundo parágrafo</p>
<span>
```

**** - Deixa o seu texto em negrito.

```
<b>Texto em Negrito</b>
```

**** - Deixa o seu texto em itálico.

```
<i>Texto em Itálico</i>
```

<hr> - Cria uma linha horizontal

```
<p>Sou um parágrafo</p>
<hr>
<p>Sou um parágrafo</p>
```

Imagem

Para colocar uma imagem na página é bem simples, vejamos:

```
<img src="nomeDoArquivo.formato">
```

Você talvez deve ter se perguntado "*o que é esse src?*". O *src* vem de source, ele é atributo da tag *<img>*, nele vai conter o caminho da imagem que será inserida por você.

Alguns pontos importantes:

- A tag img não tem a chave de fechamento.

- Se você tiver com a imagem de uma pasta, deve colocar o caminho dela dentro do src.

- Recomendamos que você coloque o atributo *alt* para que pessoas com deficiência (*através de um leitor*) saibam do que se trata a imagem na página.

```
<img  src="nomeDaPasta/nomeDoArquivo.formato"
alt="descrição sobre a imagem">
```

Link

A tag <a> define um hiperlink, que é usado para linkar de uma página a outra.

O atributo mais importante do elemento é o atributo *href*, que indica o destino do link.

```
<a href="link-do-site">Sou um link</a>
```

Div

A tag * * define uma divisão ou seção em um documento HTML. A tag * * é facilmente estilizada usando o atributo *class* ou *id*. Qualquer tipo de conteúdo pode ser colocado dentro da tag * *.

Mesmo tendo a sua funcionalidade parecida com o ** que são geralmente utilizados para guardar uma pequena informação. O * * é usados para uma divisão de um conteúdo pois, como o uso dela ajuda a quebrar os elementos em linhas, deixando melhor a visualização.

```
<div>
 <p>Sou uma div</p>
</div>
```

Listas

Vamos agora falar um pouco sobre as listas (*ordenadas e desordenadas*) e como elas funcionam no HTML. As listas são muito importantes quando queremos listar itens na página.

Listas ordenadas

As listas ordenadas (*ou numeradas*) são usadas para indicar alguma sequência ou numeração.

```
<ol>
  <li>html</li>
  <li>css</li>
  <li>javascript</li>
  <li>front-end</li>
</ol>
```

Aparecerá assim na página:

1. html
2. css
3. javascript
4. front-end

Caso você queira deixar em ordem alfabética é simples, coloque o atributo *type="a"* dentro da tag *<ol>*.

```
<ol type="a">
  <li>html</li>
  <li>css</li>
  <li>javascript</li>
  <li>front-end</li>
</ol>
```

Aparecerá assim na página:

a. html
b. css
c. javascript
d. front-end

Você também pode deixar com algarismos romanos.

```
<ol type="I">
 <li>html</li>
 <li>css</li>
 <li>javascript</li>
 <li>front-end</li>
</ol>
```

Aparecerá assim no site:

```
I. html
II. css
III. javascript
IV. front-end
```

Listas desordenadas

As listas não numeradas são usadas para listar itens, sem se preocupar com sua sequência. Chamamos de lista de marcadores apenas. Ela segue o mesmo padrão da ordenada apenas mudando de *<ol>* para *<ul>*. Em seu visual ele mudará de números para pontos.

```
<ul>
  <li>html</li>
  <li>css</li>
  <li>javascript</li>
  <li>front-end</li>
</ul>
```

Aparecerá assim no site:

- `html`
- `css`
- `javascript`
- `front-end`

Formulário

Nesta seção vamos mostrar como montar seu formulário.

Formulários HTML são um dos principais pontos de interação entre o usuário e sua página. Um formulário HTML é feito de um ou mais widgets. Esses widgets podem ser campos de texto, caixas de seleção, botões, checkboxes e radio buttons.

Para construir o nosso formulário de contato, vamos utilizar os seguintes elementos *<form>, <label>, <input>, <textarea> e <button>.*

Form

Todos formulários HTML começam com um elemento <form> como este:

```
<form action="/pagina-processa-dados-do-form"
method="post"></form>
```

O *action* especifica para onde enviar os dados do formulário quando um formulário é enviado.

O *method* especifica o método HTTP a ser usado ao enviar dados do formulário (*ele pode ser get ou post*).

Label, Input e Textarea

O nosso exemplo de formulário é muito simples e contém três campos de texto, cada um com uma etiqueta (*label*). O campo de entrada para o nome será um campo básico texto de linha única (*input*); o campo de entrada do e-mail será um campo de texto com uma única linha (*input*) que vai aceitar apenas um endereço de e-mail; o campo de entrada para a mensagem será um campo de texto de várias linhas (*textarea*).

Em termos de código HTML, teremos algo assim:

```
<form action="/pagina-processa-dados-do-form"
method="post">
    <span>
        <label>Nome:</label>
        <input type="text"/>
    </span>
    <span>
        <label>E-mail:</label>
        <input type="email"/>
    </span>
    <span>
        <label>Mensagem:</label>
        <textarea></textarea>
    </span>
</form>
```

Algumas observações:

No input temos o atributo *type*. Esse atributo define a forma que nosso input se comporta. Exemplo: o *type="email"* ele define que o campo aceita só endereço de e-mail.

Por último, mas não menos importante, a tag *<textarea>*. Ela define um controle de entrada de texto de várias linhas

geralmente usado para coletar entradas do usuário, como comentários ou revisões.

Elemento Button

O nosso formulário está quase pronto, nós temos apenas que adicionar um botão para permitir que o usuário envie os dados preenchidos no formulário. Isto é simplesmente feito usando a tag *<button>*.

```
<button type="submit">Enviar</button>
```

Semânticas

Na vida sempre temos uma forma correta de fazer as coisas, no HTML não é diferente! As tags semânticas além de deixar o código melhor para o SEO dos navegadores, ela ajuda outros desenvolvedores a entender seu código só batendo o olho, isso é ótimo para que outros desenvolvedores contribuem em seus projetos.

Uma observação:

As tags semânticas não têm nenhum efeito na apresentação na página.

Elementos semânticos: tem significado e deixam seu conteúdo claro.

Elementos não semânticos: não deixam seu conteúdo claro.

Exemplo de elementos não semânticos: * * e ** e de elementos semânticos: *<form>, <table>, <article>, <footer> e <section>.*

Veja que div é amplo, mas a tag *<footer>* dá significado (*que é o rodapé*).

Portanto ao invés de:

```
<div>Sou o rodapé</div>
```

não seria melhor:

```
<footer>Sou o rodapé</footer>
```

As principais tags semânticas

<article> - Expressa um elemento independente, ou seja, que possa ser lido e interpretado sem depender do resto da página.

<aside> - Por conceito, expressa um conteúdo a parte do conteúdo da página.

<datails> - Apresenta detalhes adicionais que o usuário pode mostrar ou esconder.

<figcaption> - Representa uma legenda para um elemento *<figure>*.

<figure> - Representa um conteúdo independente, como ilustrações, diagramas, etc...

<footer> - Representa o rodapé.

<header> - representa o cabeçalho.

<main> - Representa o conteúdo principal de um documento.

<mark> - Representa um texto destacado.

<nav> - Representa links de navegação.

<section> - Representa uma seção dentro de um documento.

<summary> - Representa um cabeçalho para um elemento *details*.

<time> - Representa uma data/hora.

Outras tags do HTML

<audio> - Inserir arquivo de áudio.

<canvas> - Área utilizada para desenhar gráficos em javascript.

<command> - Comando personalizado, pode ser chamado pelo usuário.

<datatemplate> - Bloco para templates de dados.

<embed> - Inserir conteúdo externo como SWF, vídeos, áudios.

<keygen> - Gerar pares de chaves em um formulário.

<meter> - Controle gráfico que mostra uma barra métrica que indica graficamente um valor.

<progress> - Barra de progresso, tipo loading para mostrar o progresso de uma determinada tarefa.

<source> - Apontador para os arquivos de mídia das tags <audio> e <video>.

<vídeo> - Tag para inserir vídeos na página.

<wbr> - Especifica uma quebra de linha para textos longos, especificamos onde o texto será quebrado.

Fundamentos CSS

O que é CSS?

CSS ou folhas de estilo em cascata é a linguagens de marcação (*não de programação*) responsável por adicionar estilos em nossas páginas web, como cores, tamanhos e posicionamentos. Sem ele, os sites são apenas um monte de texto e links.

Aplicando CSS

Vamos por etapas!

Etapa 1 : Criar um arquivo CSS

A primeira coisa que precisamos fazer é criar um arquivo CSS do qual nossa página HTML possa obter seu estilo. Então em seu editor de código crie um novo arquivo chamado *style.css*.

Etapa 2: Linkar seu CSS no HTML

Precisamos conectar nossa página *style.css* à página HTML. Dentro da tag <*head*> de sua página HTML, vamos adicionar uma tag <*link*> para conectar a nossa nova folha de estilo.

```
<link rel="stylesheet" href="style.css">
```

Certifique-se de que sua página HTML e sua folha de estilo CSS estão no mesmo nível de pasta, caso ele esteja dentro de uma pasta é simples chamar, basta colocar o nome da página antes do nome do arquivo separado ele com uma barra (/).

```
<link rel="stylesheet"
href="nomeDaPasta/style.css">
```

Etapa 3: Selecione e dê estilo aos elementos

Experimente selecionar um elemento e estilizá-lo!

```
h1 {
  color: #00f;
  font-size: 26px;
}
```

Isso para que todos os elementos *<h1>* em sua página HTML fique com um tamanho 26px e com a cor azul *(atualize a página sempre que aplicar um CSS ou HTML).*

Sintaxe

A sintaxe do CSS é bem simples, precisamos indicar um seletor que é uma tag, id ou classe, e dentro das chaves inserimos os comandos referente à formatação.

```
seletor {
 propriedade:valor;
}
```

Identificadores

Você deve se perguntar "*okay, mas se eu tiver mais de um elemento HTML e quiser mudar a cor do texto só de um?*".

É aí que entram os os identificadores *id* e *class.*

As classes são uma forma de identificar um grupo de elementos. Através delas, pode-se atribuir formatação a VÁRIOS elementos de uma vez

As ids são uma forma de identificar um elemento, e devem ser ÚNICAS para cada elemento. Através delas, pode-se atribuir formatação a um elemento em especial.

Para fazemos referência a uma classe usando um . (*ponto*) e o nome da classe, exemplo:

```
.souClass {
  color: #f00;
}
```

E para fazemos referência a um id usando um # e o nome do id:

```
#souID {
  color: #f00;
}
```

Dessa forma, todos os elementos que tiverem a class .souClass e o id #souID vão ficar com a cor de texto de vermelho.

E qual usar?

"*Se os dois fazem a mesma coisa, qual devo usar?*"

Agora te bateu uma dúvida né?!

A resposta é simples, se você tem vários elementos e queira usar de cor de texto vermelha em mais de uma, nesse caso usamos a class, pense a class em uma união em comum.

Vou citar um exemplo que meu professor disse na faculdade.

"*Todos nós temos uma class em comum, que no caso é humano, porém, todos nós temos um id, que no caso é o rg.*"

Então, caso queira mudar somente em um elementos, usamos o id e mais de um usamos a class.

No HTML eles são definidos dessa forma:

```
<p class="souClass">
<h1 id="souID">
```

Alguns seletores CSS

.class - Seleciona todos os elementos que usam a classe.

#id - Seleciona o elemento com o nome id.

***** - Seleciona todos os elementos.

h2 - Seleciona todos os elementos com a tag *<h2>*.

div, p - Seleciona todos os elementos * * e *<p>*.

div p - Seleciona todos os elementos *<p>* que estão dentro de elementos * *.

div > p - Seleciona todos os elementos *<p>* onde o pai seja um elemento * *.

div + p - Seleciona todos os elementos *<p>* que estão posicionados imediatamente após o elemento * *.

p ~ ul - Seleciona todos os elementos *<p>* que são irmãos de um elemento *<ul>*.

Propriedades de preenchimento

As propriedades de preenchimento CSS são usadas para gerar espaço em torno ou dentro do conteúdo.

Padding

O preenchimento limpa uma área ao redor do conteúdo (dentro da borda) de um elemento.

Margin

A propriedade magem define o tamanho do espaço em branco fora da borda.

Resumindo

Padding é o espaço entre o conteúdo e a borda, enquanto margin é o espaço fora da borda.

Como definir preenchimento

Elas podem ser aplicadas nos quatros lados de um elemento: superior, direita, inferior e esquerda (*nessa ordem*). No exemplo vamos usar o margin mas também pode ser aplicada no padding.

Margens iguais nos quatros lados do elemento:

```
.class {
 margin: 20px;
}
```

Margem superior e inferior de 5px e margem esquerda e direita de 10px:

```
.class {
 margin: 5px 10px;
}
```

Você também pode definir os espaços individualmente:

```
.class {
 margin: 0px 5px 10px 15px;
}
```

Display

Basicamente todos os elementos têm um valor padrão para sua propriedade display, a maioria dos elementos tem seu *display* configurado em *block* ou *inline-block*.

Block

O elemento *block*, não aceita elementos na mesma linha que ele, ou seja, quebra a linha após o elemento, e sua área ocupa toda a linha onde ele é inserido.

Alguns elementos que têm como padrão block: * , <h1> até <h6>, <p>, <form>, <header>, <footer>, <section>, <table>.*

Inline

O elemento inline não inicia em uma nova linha nem quebra de linha após o elemento, outro detalhe é que eles não ocupam a linha inteira, somente ocupam o espaço de seu conteúdo.

Alguns elementos que têm como padrão inline: *, <a>, <img>.*

Position

A propriedade position específica como um elemento será posicionado na tela, podemos até posicionar em um ponto específico controlando com os parâmetros top, right, bottom e left.

São quatro tipos de posicionamento disponíveis: static, relative, fixed e absolute. A única configuração que não permite escolher um posicionamento para o elemento é static.

top - Desloca o elemento na vertical (Y), o valor é a distância do elemento com o topo.

right - Desloca o elemento na horizontal (X), o valor é a distância do elemento com a borda direita.

bottom - Desloca o elemento na vertical (Y), o valor é a distância do elemento com a borda inferior.

left - Desloca o elemento na horizontal (X), o valor é a distância do elemento com a borda esquerda.

static

Este é o valor padrão de todos os elementos HTML, neste posicionamento os elementos não podem ser controlados por top, right, bottom e left, e não tem seu posicionamento afetado pelo posicionamento de outros elementos.

relative

Um elemento com relative tem seu posicionamento relacionado com o elemento anterior.

absolute

Um elemento com absolute tem seu posicionamento relacionado com o elemento pai e não com o elemento anterior, desta maneira elementos anteriores não irão afetar seu posicionamento.

fixed

O elemento com fixed tem o mesmo comportamento do absolute, só que como o nome já diz, ele fica fixo na tela, isso é, mesmo se acontecer a rolagem ele ficará fixado na página.

Fontes

Para colocar fonte em sua página é preciso chamar ele no seu HTML, para isso vamos entrar no site do Google Fonts. Logo em seguida, procure a fonte que você deseja e clique no botão "Select this style" para assim adicionar os estilos de fontes que você deseja (*perceba que quando você selecionar uma vai abrir uma aba na lateral direita*).

Nessa aba vai ter duas opções para colocar em sua página, uma é o *<link>* e o outro *@import*. O link é para importar no seu HTML e o import é para o seu CSS, agora você me pergunta “qual eu devo usar?”. Você pode escolher qualquer um, isso vai com seu gosto.

Perceba logo depois de selecionar o *<link> (na página do Google Fonts)*, em baixo tem uma código HTML e o nome font-family, é exatamente ele que vamos usar para dizer que aquela vai ser a fonte da nossa página.

No seu HTML dentro da tag *<head>* coloque o código HTML. E em seu CSS, coloque o outro código.

```
* {
  font-family: 'Nome da sua fonte aqui';
}
```

Uma observação

Você está se perguntando "*o que é esse * no CSS?*". Ele é um elemento universal, quer dizer que vai aplicar em toda sua página de estilo.

Reset

Agora vamos resetar os navegadores...mas espera, o que isso quer dizer?

Quer dizer que os navegadores tem um padrão, e alguns desses padrões não são legais para nós desenvolvedores, então vamos tirar alguns deles (*três para falar a verdade*)!

Primeiro vamos zerar os espaçamentos das nossas páginas. Quando criamos uma página HTML, por padrão, nosso site tem um espaçamento e quando criamos algo, nossos estilos não ficam da forma que queremos, eles acabam sempre tendo um espaço em volta da página. Para tirar esse padrão usamos duas coisas. Uma é o padding (*espaçamento dentro do conteúdo*) e o margin (*espaçamento fora do conteúdo*). Veja o exemplo:

```
* {
  font-family: 'Nome da sua fonte aqui';
  padding: 0;
  margin: 0;
}
```

E por último, vamos aplicar o *box-sizing: border-box;* no nosso CSS. Esse é super importante na criação dos elementos! Quando criamos um container sempre usamos uma largura e altura (*no nosso exemplo vai ser 300px em cada um*). Mas caso você queira colocar um espaçamento interno nele (*padding*) de 30px, o elemento vai deixar de ser 300px e será 330px.

Mas a gente não quer isso, não é?

Queremos que o elemento continue 300px (*na altura e largura*) porém, com um espaçamento dentro dela. É aí que entra o nosso querido *box-sizing: border-box;*

```
* {
  font-family: 'Nome da sua fonte aqui';
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}
```

Algumas unidades de medidas

em - Unidade relativa ao tamanho do font-size anterior.

rem - Unidade relativa ao font-size do elemento raiz.

vw - Unidade relativa a 1% da largura da janela.

vh - Unidade relativa a 1% da altura da janela.

% - Porcentagem.

cm - Unidade absoluta relacionada a centímetros.

px - Unidade absoluta relacionada à pixels.

pt - Unidade absoluta relacionada a pontos.