



Digital
College

FORMAÇÃO EM

DESENVOLVEDOR **FULL STACK**

UNIDADE 1:

CONCEITOS E FUNDAMENTOS WEB

MÓDULOS 2 E 3:

> **HTML 5 E CSS3** <

1.1 Introdução ao HTML e suas tags

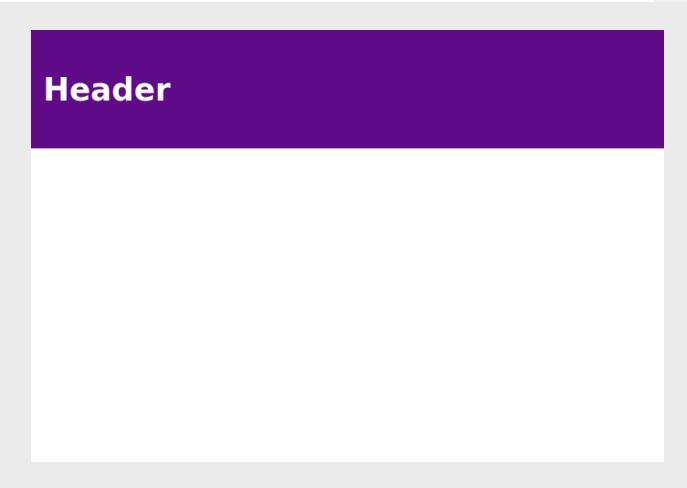
HTML é uma abreviação de *Hypertext Markup Language*, ou seja, Linguagem de Marcação de Hipertexto. Resumindo, o HTML é uma linguagem usada para a publicação de conteúdo (texto, imagens, vídeos, áudio etc.) na web.

Para que você possa entender bem, **o HTML é baseado no conceito de hipertexto, que são conjuntos de elementos ligados por conexões, que podem ser palavras, imagens, vídeos, áudio, documentos etc.** que quando conectados, formam uma grande rede de informação. A conexão feita em um hipertexto é algo imprevisto que permite a comunicação de dados, organizando conhecimentos e guardando informações relacionadas. Os elementos em “**nível de bloco**” ocupam todo o espaço do seu elemento pai (container).

<header>

Define o cabeçalho da página, geralmente no cabeçalho identificamos o site, com a logo.

Exemplo:



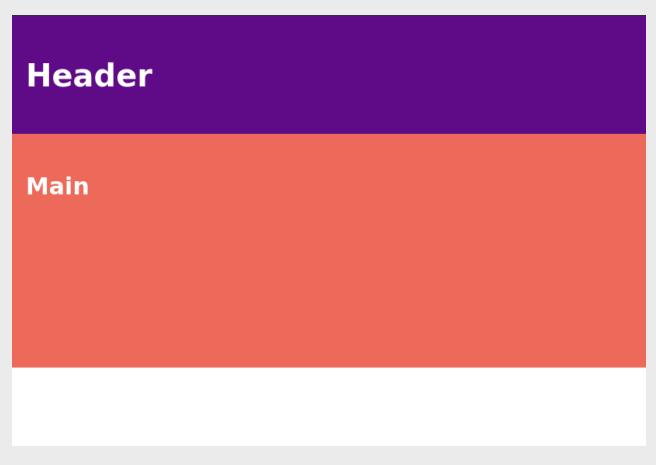
Header

A diagram showing a grey rectangular container. Inside, there is a purple horizontal bar at the top labeled "Header". Below it is a white area representing the main content of the page.

<main>

Compreende o conteúdo principal do corpo da página.

Exemplo:



Header

Main

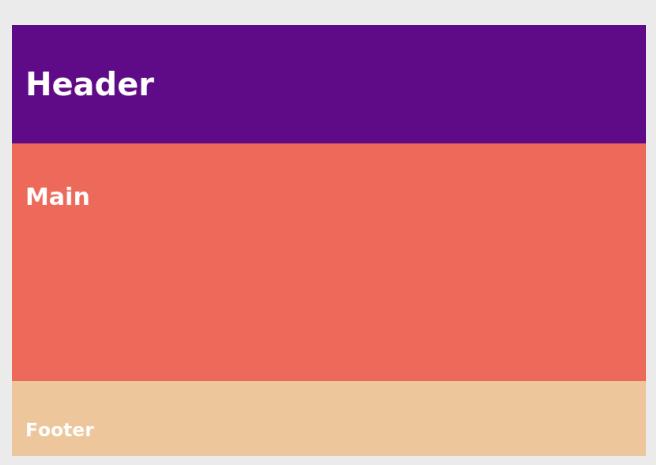
Footer

A diagram showing a grey rectangular container. It is divided into three horizontal sections: a purple top section labeled "Header", an orange middle section labeled "Main", and a white bottom section labeled "Footer".

<footer>

Define o final da página ou conteúdo, o rodapé, geralmente colocamos contatos, redes sociais, endereço, informações “institucionais” no geral.

Exemplo:



Header

Main

Footer

A diagram showing a grey rectangular container. It is divided into three horizontal sections: a purple top section labeled "Header", an orange middle section labeled "Main", and a yellow bottom section labeled "Footer".

<section>

Dentro do conteúdo principal, essa tag comprehende uma seção da página.

<article>

Inclui um artigo da página, muito utilizado em blogs e páginas de criação de conteúdo, também indica o principal conteúdo de texto da página.

<aside>

Representa uma seção que faz referência a outro conteúdo da página, como uma definição, uma explicação extra, avisos, biografia do autor, ou seja um conteúdo complementar.

<nav>

Contempla o menu de navegação das páginas do site, e dentro inserimos a listas e links com a tag .

<div>

Assim como as outras tags, também funciona como um container, porém a grande diferença é que a div não tem valor semântico, é apenas uma divisão na página para fins de layout.

<p>

Representa um parágrafo.

<h1>, <h2><h6>

A família de cabeçalhos ou headings, define os títulos da página. Esse grupo de elementos possuem um alto valor semântico e uma organização hierárquica, indo de <h1> até <h6>, sendo o h1 de maior valor semântico e o h6 de menor valor semântico.

Exemplo:



```
✓ HTML
1 <h1>Olá Mundo!</h1>
2 <h2>Olá Mundo!</h2>
3 <h3>Olá Mundo!</h3>
4 <h4>Olá Mundo!</h4>
5 <h5>Olá Mundo!</h5>
6 <h6>Olá Mundo!</h6>
```



Olá Mundo!

Olá Mundo!

<hr>

Essa tag constrói uma linha horizontal entre elementos, representa semanticamente uma quebra de conteúdo.

<video>

Utilizada para inserir vídeos no site, a tag possui atributos como width recebendo como valor a largura do vídeo em pixels, height recebendo como valor a altura video em píxeis, caso não for informada esses atributos, será utilizado a largura e altura padrão do vídeo, controls (quando presente nos permite controlar o video), src recebendo como valor o link ou diretório do arquivo de vídeo.

Exemplo:

```
<video src="" controls></video>
```

1.2 Uso do DOCTYPE e html 5

O HTML5 deixou de ser baseado no SGML, portanto, o doctype deixou de ser uma DTD que informa a versão específica do HTML. Desde então, a declaração do **Doctype serve para informar ao navegador que o documento se encontra em HTML**. Portanto, o navegador processa e manipula o documento para a versão mais atual do HTML.

Dessa forma, todas as próximas versões do HTML, como uma versão 6, 7, etc, manterão a declaração da mesma forma que funcionam no HTML5:

```
<!DOCTYPE html>
```

1.3 Tag <html>

tags <html>

As tags são usadas para informar ao navegador a estrutura do site. Ou seja: quando se escreve um código em **HTML**, as tags serão interpretadas pelo navegador, produzindo assim a estrutura e o conteúdo visual da página.

A principal característica das tags é estarem sempre dentro dos sinais de chevron (**sinal de “maior que” é “menor que”, ou seja: <>**).

As tags HTML são divididas em dois tipos: as que precisam de fechamento e as que não precisam de fechamento. As tags que precisam de fechamento possuem a sintaxe **</tag>**, já as que não precisam de fechamento possuem como estrutura **<tag/>**.

Além disso, uma mesma tag pode receber um ou mais atributos, que possuirá um valor que modifica sua estrutura ou funcionalidade.

1.4 Como definir o título e os parágrafos de um texto

Títulos, subtítulos e parágrafos em HTML são elementos extremamente comuns em quase todos os sites. O bom uso destes elementos de textos podem trazer resultados positivos para a navegação no site e a apresentação dos seu documento HTML.

A tag de título principal, é representada pelo “`<h1>`” e se encerra com “`</h1>`”, os subtítulos mantém a mesma sintaxe, seguindo a sequência que vai do “`<h2>`” ao “`<h6>`”. É recomendado o uso de um “`<h1>`” por página, como título principal.

Quando utilizamos títulos e subtítulos, podemos usar o “`hgroup`”, esse elemento agrupa os títulos facilitando a navegação dentro do código.

Veja o exemplo:

```
<body>
  <hgroup>
    <h1>Este é um título principal</h1>
    <h2>Este é um subtítulo</h2>
    <h3>Este é um subtítulo</h3>
    <h4>Este é um subtítulo</h4>
    <h5>Este é um subtítulo</h5>
    <h6>Este é um subtítulo</h6>
  </hgroup>
</body>
```

O texto que não for um título, é recomendado que esteja em um parágrafo “`<p>`”. Todo parágrafo está localizado dentro da tag “`<body>`”:

```
<!-- versão do html utilizada (html5) -->
<!DOCTYPE html>
<!-- idioma utilizado -->
<html lang="pt-br">
<!-- metadados configurações da páginas -->
<head>
    <meta charset="UTF-8">
    <title>Títulos e Parágrafos</title>
</head>
<!-- corpo da página informações visíveis para o usuário -->
<body>
    <hgroup>
        <h1>Este é um título principal</h1>
        <h2>Este é um subtítulo</h2>
    </hgroup>
    <p>Este é um parágrafo</p>
</body>
</html>
```

1.5 Utilizando as tags <h1> e <p>

A tag **H1** é usada para determinar o título da página. Conceitualmente existem 6 níveis de títulos, sendo o h1 o mais alto de todos, ou seja, ele deve ser usado para indicar o texto mais importante da página.

Exemplo:

```
<h1>Título do texto da sua página</h1>
```

• Considerações

- O comportamento padrão do H1 é negrito e tamanho da fonte xx.
- Cada página deverá ter apenas uma tag h1, já que pressupõe que cada página tenha um único título.
- Por ser uma tag de texto, ela aceita formatação por CSS das propriedades de fonte, bloco de textos, background, box model, posicionamento, borda e layout.
- Deve ser posicionada no início do texto. Não faz sentido usar esta tag no meio do texto.

A tag **p** é usada para fazer a estruturação de textos em parágrafos dentro de um documento HTML. A princípio todos os textos deverão estar dentro desta tag, exceto:

- Quando o texto for um título, nesta nova condição deverá ser usado as tags <h1> a <h6>.
- Quando o texto for uma citação, neste caso use a tag <blockquote>.
- Quando o texto for uma lista de marcadores ou lista, neste caso use e ou e .
- Quando for uma legenda, neste caso use <legend>

Exemplo de um documento com três parágrafos:

```
<html>
<head>
<title>Estruturação de parágrafos com a tag p</title>
</head>
<body>
<p class="texto">Parágrafo com uma classe associada</p>
<p id="destaque">Parágrafo com um identificador</p>
<p style="color:#ff0000">Parágrafo com formatação de CSS inline</p>
</body>
</html>
```

1.6 Negrito, utilizando a tag

O elemento HTML **** dá ao texto uma forte importância, e é tipicamente mostrado em negrito.

O elemento **** é utilizado em conteúdos que são de "grande importância", incluindo coisas de urgentes (como alertas). Podem ser sentenças que são de grande importância para toda a página, ou, você pode meramente tentar pontuar que algumas palavras são de grande importância, comparado ao conteúdo próximo.

Tipicamente, estes elementos são renderizados por padrão, usando fontes em negrito. Contudo, ele não deve ser usado para simplesmente aplicar o estilo negrito; use o CSS **font-weight** para este propósito. Use o elemento **** para chamar a atenção para certos textos sem a indicação de grande nível de importância. Use o elemento **** para marcar textos que têm necessidade de ênfase.

Outro uso aceitável para **** é denotado com o rótulo (label) de parágrafos, que representa notas ou avisos, dentro do texto da página.

• Bold vs. Strong

Muitas vezes é confuso para novos desenvolvedores porque há tantas maneiras de expressar a mesma coisa em um website renderizado. Bold e Strong são talvez um dos casos mais comuns. Porque usar **** vs ****? Você precisa digitar muito mais ao usar strong e ela produz o mesmo resultado, certo?

Talvez não; strong é um estado lógico, e bold é um estado físico. Estados lógicos separam a apresentação do conteúdo, e ao fazer isso permitem que ele seja expresso de várias maneiras diferentes. Possivelmente em vez de renderizar um texto como negrito você queira renderizá-lo vermelho, ou num tamanho diferente, ou sublinhado, ou seja lá o que for. Faz mais sentido mudar as propriedades de apresentação de strong do que bold. Isto porque bold é um estado físico; não há separação entre a apresentação e o conteúdo, e fazer com que bold faça qualquer outra coisa diferente de deixar o texto em negrito seria confuso e ilógico.

É importante notar que **** tem outros usos, quando se quer chamar atenção sem aumentar a importância.

Exemplo:

```
<p> Ao fazer x é <strong>imperativo</strong>  
que se faça y antes de prosseguir.</p>
```

1.7 Itálico, utilizando a tag ``

O elemento HTML `` marca o texto que tem ênfase. O elemento `` pode ser alinhado, com cada nível de aninhamento indicando um grau maior de ênfase.

O elemento **`` é frequentemente usado para indicar um contraste implícito ou explícito.**

Exemplo:

```
<p>
  Em HTML 5, o que anteriormente era chamado de conteúdo
  <em>block-level</em> agora é chamado de conteúdo
  <em>flow</em> .
</p>
<i> vs <em>
```

É frequentemente confuso para novos desenvolvedores porque há vários elementos para expressar a ênfase em algum texto. **`<i>` e ``** são talvez uns dos mais comuns. Por que usar `` vs `<i></i>`? Eles produzem o mesmo resultado, certo?

Não exatamente. O resultado visual é, por padrão, o mesmo - ambas as tags renderizam seu conteúdo em itálicos. Mas o significado semântico é diferente. A tag `` representa ênfase importante de seus conteúdos, enquanto que a tag `<i>` representa o texto que é iniciado de uma prosa, como uma palavra estrangeira, pensamentos de um personagem fictional, ou quando o texto se refere à definição de uma palavra em vez de representar seu significado semântico. (O título de um trabalho, tal como o nome de um livro ou filme, deve usar `<cite>`.)

Um exemplo para `` poderia ser: "Apenas já faça isso!", ou: "Nós temos que fazer algo acerca disso". Uma pessoa ou software lendo o texto pronuncia as palavras em itálico com uma ênfase.

Um exemplo para `<i>` poderia ser: "A Rainha Mary velejou na noite passada". Aqui, não é adicionada ênfase ou importância na palavra "Rainha Mary". É meramente indicado que o objeto em questão não é uma rainha chamada Mary, mas um navio chamado Rainha Mary. Outro exemplo para `<i>` poderia ser: "A palavra o é um artigo".

1.8 Passando dados para o navegador lang, charset, title

O atributo global **lang** ajuda a definir o idioma de um elemento: a língua em que elementos não-editáveis são escritos, ou a língua em que elementos editáveis devem ser escritos pelo usuário.

Se o valor do atributo é uma string vazia (`lang=""`), o idioma é definido como `unknown` (desconhecido); se a tag de idioma não é válida conforme o BCP47, ela é definida como `invalid` (inválida).

Exemplo:

```
<p lang="en-GB">This paragraph is defined as British English.</p>
<p lang="fr">Ce paragraphe est défini en français.</p>
```

Para exibir uma página HTML corretamente, um navegador da Web deve conhecer o conjunto de caracteres usado na página.

Isso é especificado na `<meta>` tag:

```
<meta charset="UTF-8">
```

A especificação HTML5 incentiva os desenvolvedores da Web a usar o conjunto de caracteres **UTF-8**, que abrange quase todos os caracteres e símbolos do mundo!

O elemento HTML **<title>** (Elemento HTML Título) define o título do documento, mostrado na barra de título de um navegador ou na aba da página. Pode conter somente texto e quaisquer marcações contidas no texto não são interpretadas.

O elemento `<title>` é sempre usado dentro da `<head>` da página.

Exemplo:

```
<title>Título incrível</title>
```

1.9 Head e Body

A tag **head** faz parte da estrutura básica do documento, sua finalidade é definir o cabeçalho do documento com informações que não serão exibidas dentro do conteúdo da página.

A tag **<head>** possui fechamento **</head>** e poderá suportar as tags **<base>**, **<link>**, **<meta>**, **<script>**, **<style>** e **<title>** dentro da sua estrutura., como no exemplo seguinte:

```
<html lang="en">
<head>
<title>...conteúdo de title...</title>
<base ...conteúdo de base... />
<link ...conteúdo de link... />
<script>...conteúdo de script...</script>
<style>...conteúdo de style...</style>
<meta ...conteúdo de meta... />
</head>
<body>
<!-- Aqui será colocado as demais tag do documento --&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

A tag **body** faz parte da estrutura básica do documento, sua finalidade é definir o corpo do documento, ou seja, tudo que estiver dentro da tag body será mostrado de alguma forma no conteúdo da página.

A tag **<body>** possui fechamento **</body>** e na sua estrutura deverá ficar todas as demais tags, exceto as tags de cabeçalho visto no post da tag **<head>**.

Atributos

A tag body possui os atributos: **alink**, **link**, **text**, **vlink**, **bgcolor** e **background**, porém, estes atributos estão depreciados e devem ser substituídos pelas propriedades do CSS.

2.0 Hora da Prática

Utilizando os conhecimentos da matéria anterior vamos construir nossa primeira página em HTML5.

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
  </head>
  <body>
    <h1> Hello World! </h1>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
      Eos labore nesciunt ut? Nisi suscipit tempora impedit accusamus
      provident voluptatum aspernatur repellendus? Aspernatur et dolor
      fuga architecto laborum dolorem, obcaecati facilis?
    </p>
  </body>
</html>
```

3.1 CSS Inline

O **CSS inline** é usado para dar estilo a um elemento HTML específico. Para este estilo de CSS você somente vai precisar adicionar o atributo *style* para cada tag HTML, sem usar os seletores.

Este tipo de CSS não é realmente recomendado, já que cada tag HTML precisa ser estilizada de maneira individual. Gerenciar o seu site pode se tornar uma tarefa bem difícil de você só usar o CSS inline.

Contudo, o CSS inline no HTML pode ser útil para algumas situações. Por exemplo, em casos onde você não tem acesso aos arquivos CSS ou precisa aplicar estilos para um único elemento.

Vamos dar uma olhada num exemplo. Aqui, nós adicionamos um CSS inline para as tags <p> e <h1>:

Exemplo:

```
<!DOCTYPE html>
<html>
<body style="background-color:black;">

<h1 style="color:white;padding:30px;">Digital College</h1>
<p style="color:white;">Hello World!.</p>

</body>
</html>
```

• Vantagens do CSS Inline:

- Você pode inserir elementos CSS de maneira fácil e rápida numa página HTML. É por isso que esse método é útil para testar e pré-visualizar mudanças, assim como executar correções rápidas no seu site.
- Você não precisa criar e fazer *upload* de um documento separado como no estilo externo.

• Desvantagens do Inline CSS:

- Adicionar regras CSS para cada elemento HTML consome muito tempo e faz a sua estrutura HTML ficar bagunçada.
- Estilizar múltiplos elementos pode afetar o tamanho da sua página e o tempo para download.

3.2 Style, font-size, text-align

Cascading Style Sheet é usado para estilizar elementos escritos em uma linguagem de marcação como HTML. O CSS separa o conteúdo da representação visual do site. Pense na decoração da sua página. Utilizando o CSS é possível alterar a cor do texto e do fundo, fonte e espaçamento entre parágrafos. Também pode criar tabelas, usar variações de *layouts*, ajustar imagens para suas respectivas telas e assim por diante.

CSS foi desenvolvido pelo W3C (**World Wide Web Consortium**) em 1996, por uma razão bem simples. O HTML não foi projetado para ter tags que ajudariam a formatar a página. Você deveria apenas escrever a marcação para o site.

CSS não é tecnicamente uma necessidade, mas provavelmente você não gostaria de olhar para um site que usa apenas HTML, pois isso pareceria completamente abandonado.

A propriedade **font-size** no CSS estabelece o tamanho da fonte. Esta propriedade também é usada para computar o tamanho de em, ex, e outras unidades <length> (**en-US**) relativas.

Exemplo:

```
font-size: 1.2em;  
font-size: x-small;  
font-size: smaller;  
font-size: 12px;  
font-size: 80%;
```

A propriedade CSS *text-align* descreve como conteúdo *inline*, como texto, é alinhado no elemento pai em bloco. *Text-align* não controla o alinhamento de elementos em bloco, apenas o seu conteúdo *inline*.

Exemplo:

```
/* Keyword values */  
text-align: left;  
text-align: right;  
text-align: center;  
text-align: justify;  
text-align: justify-all;  
text-align: start;  
text-align: end;  
text-align: match-parent;  
  
/* Block alignment values (Non-standard syntax) */  
text-align: -moz-center;  
text-align: -webkit-center;  
  
/* Global values */  
text-align: inherit;  
text-align: initial;  
text-align: unset;
```

3.3 Tag style

O elemento HTML **<style>** contém informações de estilo para um documento ou uma parte do documento. As informações de estilo específico estão contidas dentro deste elemento, geralmente no **CSS**.

Exemplo uma folha de estilo simples:

```
<style type="text/css">
body {
  color:red;
}
</style>
```

3.4 style.css

Em "**Começando com o CSS**" nós linkamos uma folha de estilos externos em nossa página. Esse é o método mais comum utilizado para juntar CSS em um documento, podendo utilizar tal método em múltiplas páginas, permitindo você estilizar todas as páginas como a mesma folha de estilos. Na maioria dos casos, as diferentes páginas do site vão parecer bem iguais entre si e por isso você pode usar as mesmas regras para o estilo padrão da página.

Uma folha de estilos externa é quando você tem seu CSS escrito em um arquivo separado com uma extensão **.css**, e você o refere dentro de um elemento <link> do HTML:

Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

O arquivo CSS deve se parecer com algo nesse estilo:

```
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

p {
  color: red;
}
```

O atributo href do elemento <link>, precisa fazer referência a um arquivo em nosso sistema de arquivos.

No exemplo abaixo, o arquivo CSS está na mesma pasta que o documento HTML, mas você pode colocá-lo em outro lugar e reajustar o caminho marcado para encontrá-lo, como a seguir:

Exemplo:

```
<!-- Inside a subdirectory called styles inside the current directory -->
<link rel="stylesheet" href="styles/style.css">

<!-- Inside a subdirectory called general, which is in a subdirectory called
styles, inside the current directory -->
<link rel="stylesheet" href="styles/general/style.css">

<!-- Go up one directory level, then inside a subdirectory called styles -->
<link rel="stylesheet" href="../styles/style.css">
```

3.5 Alterando as cores de parágrafos, background e fontes

Os valores de cores podem ser expressos como palavras-chave de cores, números de cores hexadecimais ou números de cores RGB. Para esta lição, você precisará ter um documento HTML para ver as alterações CSS e um arquivo CSS separado anexado a esse documento. Vamos dar uma olhada no elemento de parágrafo, especificamente.

Para alterar a cor do texto para cada parágrafo em seu arquivo HTML, vá para a folha de estilo externa e digite **p {}**. Coloque o cor propriedade no estilo seguido por dois pontos, como **p {cor:}**. Em seguida, adicione o valor da sua cor após a propriedade, terminando com um ponto e vírgula. Neste exemplo, o texto do parágrafo é alterado para a cor preta.

Exemplo:

```
p {  
  
color:#000;  
  
}
```

Utilizamos essa propriedade para **alterar a cor do plano de fundo**, seja de um elemento ou do documento como um todo. Para isso, basta atribuir a cor desejada à propriedade background-color, declarada dentro do seletor no qual ela deve ser aplicada.

Exemplo de uso:

```
/* Plano de fundo em vermelho e opacidade de 50% */  
background-color: rgba(255,0,0,0.5);
```

Nesse exemplo utilizamos a função RGBA das CSS para definir o vermelho com uma transparência como cor de fundo.

3.6 Elementos pai e filhos

Pseudo-classes permitem selecionar elementos baseados em informações que não estão contidas na árvore de documentos como um estado ou que é particularmente complexa de extrair. Por exemplo, eles correspondem se um link foi visitado anteriormente ou não.

Pseudo-elementos são abstrações da árvore que representam entidades além do que o HTML faz. Por exemplo, o HTML não tem um elemento que descreve a primeira letra ou linha de um parágrafo, ou o marcador de uma lista. Os pseudo-elementos representam essas entidades e permitem que as regras CSS sejam associadas a elas. Desta forma, essas entidades podem ser denominadas independentemente.

- **Bold vs. Strong**

- **Vantagens do CSS Inline:**

Este seletor básico escolhe todos os elementos que correspondem ao nome fornecido.

Sintaxe:

nome-da-tag

Exemplo:

input corresponderá a qualquer elemento <input>.

- **Seletor por classe**

Este seletor básico escolhe elementos baseados no valor de seu atributo classe. Sintaxe: .nome-da-classe

Exemplo:

.index irá corresponder a qualquer elemento que tenha o índice de classe (provavelmente definido por um atributo class="index", ou similar).

- **Seletor por ID**

Este seletor básico escolhe nós baseados no valor do atributo id. Deve existir apenas um elemento com o mesmo ID no mesmo documento.

Sintaxe:

#nome-do-id

Exemplo:

#toc irá corresponder ao elemento que possuir o id=toc (definido por um atributo id="toc", ou similar).

- **Seletores universais**

Este seletor básico irá escolher todos os nós. Ele também existe em um namespace único e em uma variante de todo o namespace também.

Sintaxe:

* ns|* *|*

Exemplo:

* Irá corresponder a todos os elementos do documento.

• Seletores por atributo

Este seletor básico irá escolher nós baseados no valor de um de seus atributos, ou até mesmo pelo próprio atributo.

Sintaxe:

[atrib] [atrib=valor] [atrib~=valor] [atrib|=valor]
[atrib^=valor] [atrib\$=valor] [atrib*=valor]

Exemplo:

[autoplay] irá corresponder a todos os elementos que possuírem o atributo autoplay (para qualquer valor).

• Seletores de irmãos adjacentes

O combinador + seleciona os nós que seguem imediatamente o elemento especificado anteriormente.

Sintaxe:

A + B

Exemplo:

ul + li irá corresponder a qualquer elemento **** que segue imediatamente após um elemento ****.

• Seletores gerais de irmãos

O combinador ~ seleciona os nós que seguem (não necessariamente imediatamente) o elemento especificado anteriormente, se ambos os elementos compartilham o mesmo pai.

Sintaxe:

A ~ B

Exemplo:

p ~ span irá corresponder a todo elemento **** que seguir um elemento **<p>** dentro de um mesmo elemento pai.

• Seletor de filhos:

O combinador > seleciona nós que são filhos diretos do elemento especificado anteriormente.

Sintaxe:

A > B

Exemplo:

ul > li irá corresponder a todo elemento **** que estiver diretamente dentro de um elemento **** especificado.

• Seletor de descendentes

O combinador seleciona os nós que são filhos do elemento especificado anteriormente (não é necessário que seja um filho direto).

Sintaxe:

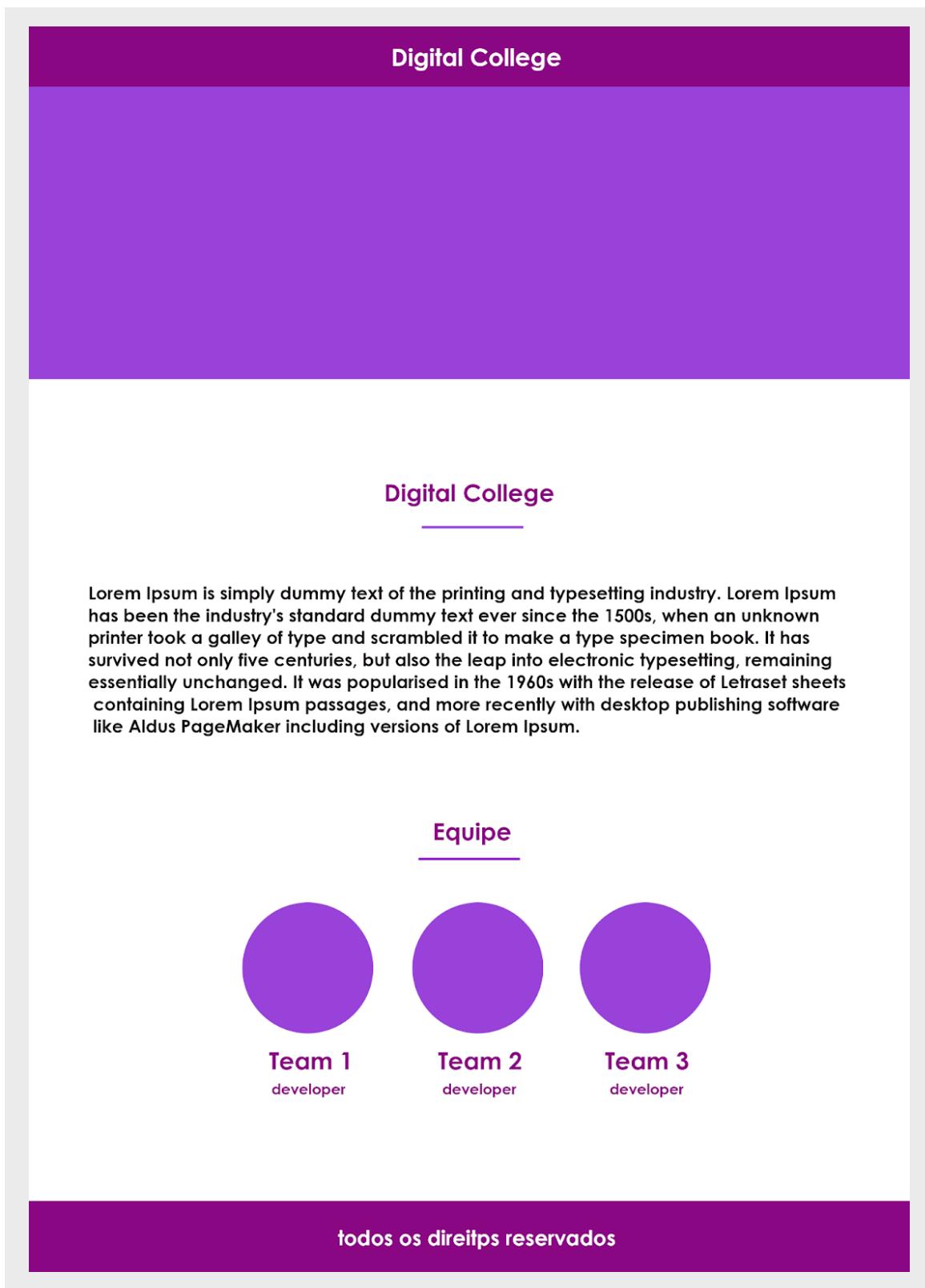
A B

Exemplo:

div span irá corresponder a todo e qualquer elemento **** que estiver dentro do elemento **<div>**.

4.0 Prática

Utilizando a referência abaixo vamos produzir nossa primeira página utilizando html e css.



The wireframe illustrates a website layout:

- Header:** A purple header bar at the top contains the "Digital College" logo.
- Main Content Area:** The main content area is white and features a title "Digital College" with a horizontal line underneath it. Below the title is a paragraph of placeholder text (Lorem Ipsum) describing its history in the printing industry.
- Sidebar:** A sidebar on the right side contains the word "Equipe" above three circular icons, each labeled "Team 1 developer", "Team 2 developer", and "Team 3 developer".
- Footer:** A purple footer bar at the bottom contains the text "todos os direitos reservados".

5.0 Inserir imagem

O elemento HTML `` (or HTML Image Element) representa a inserção de imagem no documento, sendo implementado também pelo HTML5 para uma melhor experiência com o elemento `<figure>` e `<figcaption>`.

Exemplo:

```
<figure>
  
  <figcaption> Informações da Figura </figcaption>
</figure>
```

• Atributo ALT

Este atributo define um texto alternativo que descreve a imagem. Os Usuários irão ver o texto se a URL da imagem estiver errada, a imagem não está em um dos formatos suportados ou até a imagem ser baixada .

• Atributo SRC

URL da imagem, este atributo é obrigatório para o `` elemento. Em navegadores que suportam `srcset` , `src` é ignorado se este for fornecido.

5.1 CSS aplicado em imagens

Qualquer edição de imagem não primitiva requer mais que um único elemento. Isso pode ser obtido com pseudo elementos. Infelizmente, há mais uma complicação. O elemento img vem abaixo dos elementos substituídos. Assim, **:before** e **:after** não funcionarão com tags de imagem. Para resolver isso, precisaremos um invólucro para nossa imagem e a tag figure é a melhor candidata para o caso. Assim, a marcação será:

Exemplo de uso:

```
<figure>
  
</figure>
```

Todos os efeitos de edição criados terão o mesmo CSS base.

```
figure {
  position: relative;
}

figure:before,
figure:after {
  content: '';
  height: 100%;
  width: 100%;
  top: 0;
  left: 0;
  position: absolute;
}

img {
  width: 100%;
  height: 100%;
  margin: 0;
  padding: 0;
}
```

Usamos :before e :after para podermos aplicar os modos de mesclagem. Notemos que aplicamos 100% a width e height para poder cobrir figure, e usamos posicionamento absoluto para alinhamento perfeito.

Na maioria dos casos, aplicamos os filtros nas imagens usando modos de mesclagem e outros efeitos nos pseudo elementos. A imagem abaixo é a original que usaremos.

Exemplo de efeito com CSS:**• Imagens em Tons de Cinza de Alto Contraste**

Para criar uma imagem de alto contraste, podemos apenas usar um valor alto no contraste. Mas aumentar o brilho deixa o efeito dramático. Se usarmos apenas filtros, é só isso a fazer. Contudo, com modos de mesclagem, podemos adicionar uma sombra interna com mesclagem overlay à imagem para melhor resultado.

Eis o CSS para o efeito:

```
img {  
    filter: contrast(1.8) brightness(1.5) grayscale(1);  
}  
  
figure:before {  
    z-index: 3;  
    mix-blend-mode: overlay;  
    box-shadow: 0 0 200px black inset;  
}
```

Adicionar **z-index** mantém os pseudo elementos acima da imagem. Usamos o modo de mesclagem overlay para manter a imagem levemente mais escura após aplicar o box-shadow.

Passe o mouse por cima da imagem abaixo e veja a diferença entre filtros contra um combo de filtros e modos de mesclagem.

Para praticar, podemos tentar valores diferentes em **box-shadow**.

5.2 Dimensões das imagens

Se não especificamos nada na tag img, apenas sua fonte (**src**), então o navegador foi lá e carregou a foto no seu site, exatamente do jeito que ela era.

Porém, a função do HTML é, basicamente, formatar e decidir a maneira como as coisas são exibidas em um site.

A sintaxe desses atributos é:

```

```

Vamos, agora, usar tanto a altura como largura em pixels.

Por exemplo, a imagem "**HTML-Progressivo.png**" a seguir tem **388** pixels de largura e **66** pixels de altura:

HTML Progressivo

Para exibir essa mensagem com 300 pixels de largura e 50 pixels de altura, usamos a tag **img** da seguinte maneira:

```

```

E a ordem dos atributos **height** e **width** não importa! Poderíamos ter feito:

```

```

E uma das coisas que podemos comandar é a imagem, seu tamanho, altura e onde vai ficar dentro de um site
height, em inglês, significa altura.

E width, como é de se esperar, significa largura. E essas duas palavrinhas em inglês são também os atributos que iremos usar na tag img para poder definir a altura e largura de qualquer imagem.

Mesmo que, em algumas ocasiões, você vá querer usar a imagem com seu tamanho original (podendo portanto não fornecer nenhum dos atributos height ou width), recomenda-se usar tais atributos de tamanho.

O motivo disso é que ao definir a altura e largura de uma figura em seu site, aquele espaço na página ficará já reservado para a figura. Assim, enquanto o browser acessa a imagem ela é carregada em sua posição, evitando que texto e outros elementos fiquem mudando de posição à medida que uma nova imagem é carregada.

• Cuidados com height e width

Devemos tomar alguns cuidados sempre que formos usar os tamanhos de uma imagem através dos atributos height e width. O principal deles, é a proporção entre a largura e a altura de uma imagem.

Por exemplo, se uma imagem tem os tamanhos 800x600 (altura x largura ou width x height), se reduzirmos ela à metade ficará com o tamanho 400x300.

Se dobrarmos, ficará 1600x1200.

E se fizessem eater 900x100 ? Ou 2112 x 1221 ?

Teste! Veja como fica a imagem: horrível! Parece que foi esticada para os lados ou cresceu.

Se vamos multiplicar a altura por um número (2 para dobrar e 0,5 para reduzir a metade, por exemplo), devemos multiplicar esse mesmo número na largura. Isso manterá sempre a proporção e não fará que sua imagem perca a qualidade.

6.0 Hora da Prática



Utilizando a imagem de sua preferência utilize classes css para manipular a imagem.

7.1 Listas no HTML (ordenadas e não ordenadas)

Lista é um importante recurso de HTML, pois permite criarmos tópicos de textos para uma melhor exemplificação de um determinado assunto. São recursos extremamente usados, inclusive quando nem imaginamos que ele está sendo usado, como no caso de menus. Hoje, boa parte dos menus em HTML é feito com listas.

• Listas não ordenadas

As **listas não numeradas** são usadas para listar itens, sem se preocupar com sua sequência. Chamamos de lista de marcadores apenas.

As tags usadas para criar uma lista não ordenada são **** e ****. A tag **** é usada para definir a lista e a tag **** é usada para cada item da lista.

Exemplos de uso:

```
<ul>
    <li> Internet Explorer </li>
    <li> Opera </li>
    <li> Firefox </li>
    <li> Safari </li>
</ul>
```

• Listas ordenadas ou numeradas

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

As tags usadas para criar uma lista ordenada são **** e ****. A tag **** é usada para definir a lista e a tag **** é usada para cada item da lista.

Exemplos de uso:

```
<ol>
    <li> São Paulo </li>
    <li> Rio de Janeiro </li>
    <li> Belo Horizonte </li>
    <li> Brasília </li>
</ol>
```

Exemplos de utilizar listas:

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <title> Listas ordenadas, não ordenadas </title>
    </head>
    <body>
        <h2>Principais browsers</h2>
        <ul>
            <li> Internet Explorer </li>
            <li> Opera </li>
            <li> Firefox </li>
            <li> Safari </li>
        </ul>
        <h2> Principais cidades do Brasil </h2>
        <ol>
            <li> São Paulo </li>
            <li> Rio de Janeiro </li>
            <li> Belo Horizonte </li>
            <li> Brasília </li>
        </ol>
    </body>
</html>
```

7.2 Divs (div)

O **elemento de divisão** HTML **<div>** é um container genérico para conteúdo de fluxo, que de certa forma não representa nada. Ele pode ser utilizado para agrupar elementos para fins de estilos (**usando class ou id**), ou porque eles compartilham valores de atributos, como lang. Ele deve ser utilizado somente quando não tiver outro elemento de semântica (tal como **<article>** ou **<nav>**).

Exemplo de uso:

```
<div>
  <p> Qualquer tipo de conteúdo aqui. Você dá o nome! </p>
</div>
```

7.3 Span

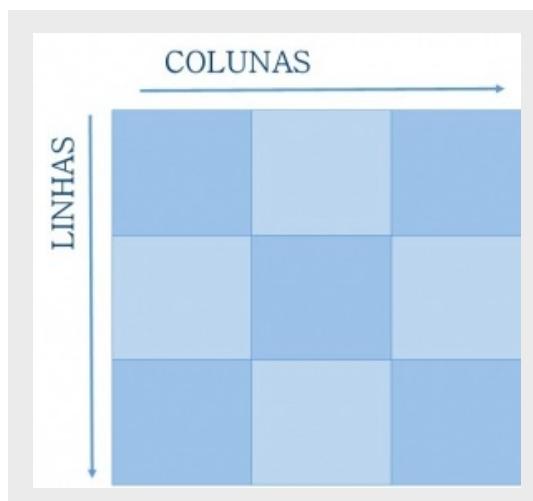
O elemento HTML **** é um contêiner genérico em linha para conteúdo fraseado, que não representa nada por natureza. Ele pode ser usado para agrupar elementos para fins de estilo (**usando os atributos class ou id**), ou para compartilhar valores de atributos como lang. Ele deve ser usado somente quando nenhum outro elemento semântico for apropriado. **** é muito parecido com o elemento **<div>**, entretanto **<div>** é um elemento de nível de bloco enquanto **** é um elemento em linha.

Exemplo de uso:

```
<p>
  <span> Apenas um texto! </span>
</p>
```

7.4 Tabelas

As tabelas são listas de dados em duas dimensões e são compostas por linhas e colunas. Portanto, são muito utilizadas para apresentar dados de uma forma organizada. Um grande exemplo de como as tabelas são importantes é o crescente uso do programa Excel, requisito obrigatório de seleção para muitas empresas. Mas e para apresentar uma tabela no seu site, como funciona? Para isso, existe a possibilidade de criar tabelas através do próprio HTML, de forma rápida e simples, por meio de uma tabela HTML.



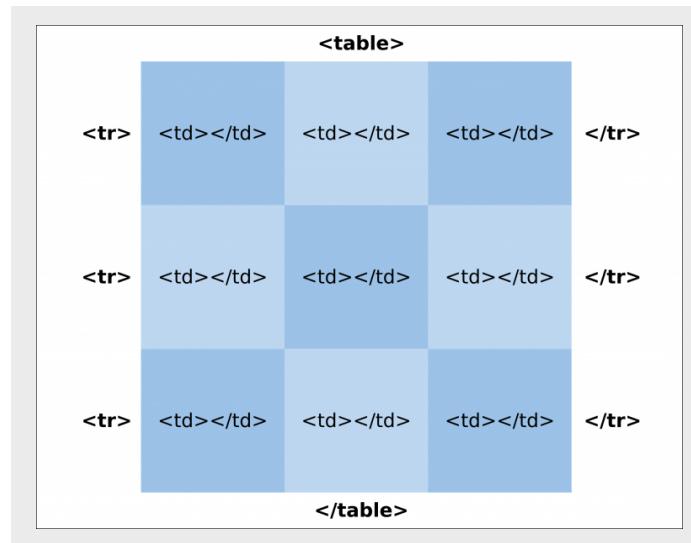
Conforme a imagem acima, podemos ver a estrutura básica de uma tabela. Assim, é importante saber que cada elemento de uma tabela é denominado “célula”.

Por exemplo, vejamos abaixo um exemplo de tabela com nove células organizadas em três colunas e três linhas, sendo a primeira linha o título das colunas:

Nome	Idade	Profissão
Albert	27	Escritor
Jim	57	Autor

A tag utilizada para criar uma tabela HTML é a tag **<table>**, posteriormente fechada com **</table>**. Dentro dela, incluímos todos os elementos que compõem nossa tabela, ou seja, as células da tabela. Assim sendo, os elementos de uma tabela consistem em outras tags que poderão ser utilizadas. Nos próximos tópicos, veremos como utilizar cada uma delas.

As tags que vão formar a **estrutura básica de uma tabela em HTML** são as tags <tr> e <td>. A tag <tr> representa uma linha e a tag <td> representa uma célula. Desta forma, a criação de colunas em uma tabela HTML é feita automaticamente através da quantidade de células incluídas dentro de uma linha. Por exemplo, vejamos na imagem abaixo como essa estrutura é formada:



Assim sendo, vamos criar nossa primeira tabela HTML com o exemplo a seguir:

```

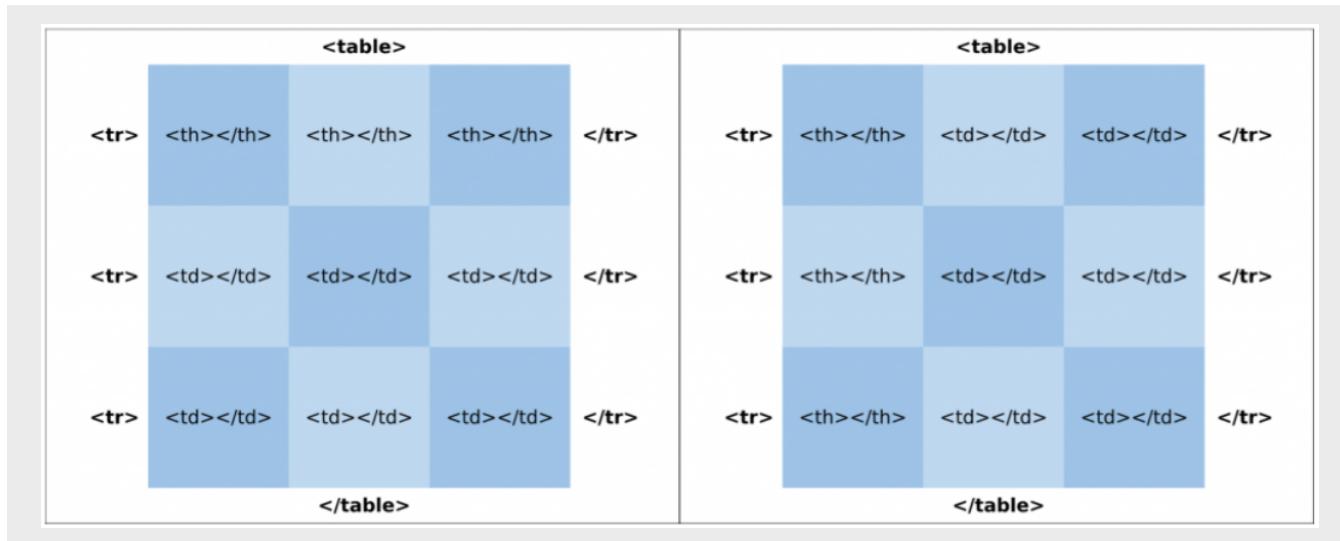
<table border="1">
  <tr>
    <td>Nome</td>
    <td>Idade</td>
    <td>Profissão</td>
  </tr>
  <tr>
    <td>Ted</td>
    <td>22</td>
    <td>Estudante</td>
  </tr>
  <tr>
    <td>Ralf</td>
    <td>26</td>
    <td>Designer</td>
  </tr>
</table>
    
```

Dessa forma, o resultado do código acima será a seguinte tabela:

Nome	Idade	Profissão
Ted	22	Estudante
Ralf	26	Designer

• Célula de título da tabela HTML

Caso queira incluir uma ou mais células que representam títulos, ganhando um destaque em relação às demais células, pode utilizar a tag <th>.



Então, a partir do exemplo anteriormente apresentado, vamos modificar a primeira linha para que todas as células desta linha sejam do tipo título:

```
<table border="1">
  <tr>
    <th>Nome</th>
    <th>Idade</th>
    <th>Profissão</th>
  </tr>
  <tr>
    <td>Ted</td>
    <td>22</td>
    <td>Estudante</td>
  </tr>
  <tr>
    <td>Ralf</td>
    <td>26</td>
    <td>Designer</td>
  </tr>
</table>
```

Dessa forma, o resultado do código acima será a seguinte tabela:

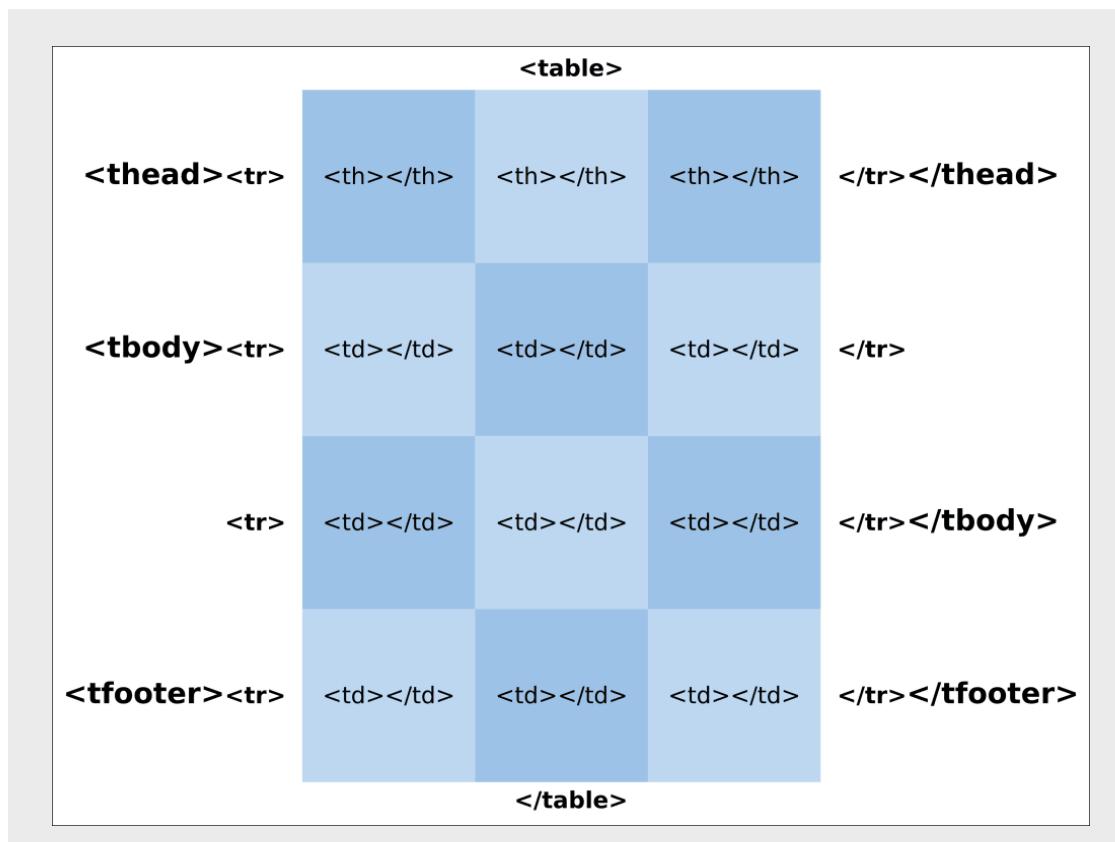
Nome	Idade	Profissão
Ted	22	Estudante
Ralf	26	Designer

- **Tags estruturais da tabela**

Anteriormente, vimos que a estrutura básica de uma tabela consiste nas tags `<tr>`, `<td>` e as células de título `<th>`. Porém, além dessas, também há tags que servem para estruturar melhor a tabela. São elas:

1. **<thead></thead>** – Esta tag representa o cabeçalho da tabela, geralmente composta por células título;
2. **<tbody></tbody>** – Essa tag representa o corpo da tabela;
3. **<tfoot></tfoot>** – Essa tag representa o rodapé da tabela.

Portanto, essas tags tem como principal efeito manter uma semântica adequada para suas tabelas. Sendo assim, vejamos então a seguinte imagem para exemplificar a estrutura de uma tabela:



Em seguida, vamos agora criar o seguinte código, utilizando como base a estrutura anterior:

```


| Título 1     | Título 2     | Título 3     |
|--------------|--------------|--------------|
| Body1 linha1 | Body2 linha1 | Body3 linha1 |
| Body1 linha2 | Body2 linha2 | Body3 linha2 |
| Foot 1       | Foot 2       | Foot 3       |


```

Com isso, chegamos ao seguinte resultado:

Título 1	Título 2	Título 3
Body1 linha1	Body2 linha1	Body3 linha1
Body1 linha2	Body2 linha2	Body3 linha2
Foot 1	Foot 2	Foot 3

Porém, vale ressaltar que nem todos os navegadores compreendem direito essas tags, em especial a tag **<tfoot>**. Portanto, vale manter as tags **<thead>** e **<tbody>**, porém evite utilizar o **<tfoot>** a menos que tenha um objetivo específico.

• Criando células vazias

Caso seja necessário incluir uma célula vazia (ou seja: sem nenhum valor dentro) na sua tabela, uma saída seria apenas incluir a **<td>** ou **<th>** sem nenhum conteúdo. No entanto, isso não é recomendado, porque alguns navegadores não conseguem ler adequadamente uma célula vazia. Portanto, a solução mais adequada para isso é utilizar **a técnica de espaço, incluindo a entidade ** conforme o exemplo abaixo:

```
<table>
  <tr>
    <th>&nbsp;</th>
    <th>Valor</th>
  </tr>
  <tr>
    <td>Item 1</td>
    <td>R$ 10,00</td>
  </tr>
  <tr>
    <td>Item 2</td>
    <td>R$ 20,00</td>
  </tr>
</table>
```

Por consequência, o resultado será a tabela a seguir:

	Valor
Item 1	R\$ 10,00
Item 2	R\$ 20,00

- **Células que abrangem mais que uma linha/coluna**

Em algumas situações, temos a necessidade de utilizar uma mesclagem de célula, ou seja, criar uma célula que abrange mais do que uma linha ou coluna. Dessa forma, podemos utilizar os atributos colspan e rowspan. O colspan="x" vai fazer uma mesclagem de colunas, e o rowspan="x" vai mesclar linhas, bastando substituir o x pelo número de colunas ou linhas que deseja ocupar. Além disso, podemos também mesclar os dois em uma mesma célula. Com isso, vamos ver os exemplos a seguir:

Primeiro, utilizando o colspan:

```
<table border="1">
  <tr>
    <th>Nome</th>
    <th colspan="2">Telefones</th>
  </tr>
  <tr>
    <td>Ted</td>
    <td>8888-8888</td>
    <td>9999-9999</td>
  </tr>
  <tr>
    <td>Junior</td>
    <td>1111-1111</td>
    <td>2222-2222</td>
  </tr>
</table>
```

Sendo assim, com o exemplo acima, obtemos a seguinte tabela HTML:

Nomes	Telefones	
Ted	8888-8888	9999-9999
Junior	1111-1111	2222-2222

Depois, utilizando o rowspan:

```
<table border="1">
<tr>
  <th>Primeiro Nome:</th>
  <td>Ted</td>
</tr>
<tr>
  <th rowspan="2">Telefone:</th>
  <td>8888-8888</td>
</tr>
<tr>
  <td>9999-9999</td>
</tr>
</table>
```

Com o exemplo acima, obtemos a seguinte tabela HTML:

Primeiro nome:	Ted
Telefone	8888-8888
	9999-9999

- **Título da tabela**

A tabela HTML tem uma tag particular para definir um título. Essa tag é a `<caption></caption>`. Em contraste com os demais elementos da tabela, a tag `<caption>` não será composta por células, portanto ficará acima da tabela, como um título.

Desta forma, veja então o exemplo a seguir:

```
<table border="1">
  <caption>Exemplo de Título</caption>
  <tr>
    <th>Nome</th>
    <th>Sobrenome</th>
  </tr>
  <tr>
    <td>Ted</td>
    <td>Junior</td>
  </tr>
  <tr>
    <td>Ted</td>
    <td>Neto</td>
  </tr>
</table>
```

Com isso, o resultado será a seguinte tabela com título:

Nome	Sobrenome
Ted	Junior
Ted	Neto

• Utilizando o `colgroup` e `col`

Anteriormente, explicamos que hoje em dia, boa parte dos atributos de uma tabela se encontra depreciado. Portanto, a estilização deve ser feita através do CSS. Dessa forma, existe um elemento muito importante que nos permite criar um grupo de colunas, podendo assim estilizar diversas colunas de uma só vez. Esse elemento é a tag `<colgroup></colgroup>`. Posteriormente, dentro do grupo de colunas, incluímos então as tags `<col>` para poder definir cada coluna.

Vejamos então o seguinte exemplo, aplicando o CSS às classes que utilizaremos nas `<col>`:

```
<style>
    .vermelho{
        background-color: red;
    }
    .amarelo{
        background-color: yellow;
    }
    .azul{
        background-color: blue;
    }
</style>

<table border="1">
    <colgroup>
        <col class="vermelho">
        <col class="amarelo">
        <col class="azul">
    </colgroup>
    <tr>
        <th>Nome</th>
        <th>Idade</th>
        <th>Profissão</th>
    </tr>
    <tr>
        <td>Ted</td>
        <td>22</td>
        <td>Estudante</td>
    </tr>
    <tr>
        <td>Junior</td>
        <td>26</td>
        <td>Escritor</td>
    </tr>
</table>
```

Com isso, o resultado deste código será:

Nome	Idade	Profissão
Ted	22	Estudante
Junior	26	Escritor

Além dessa forma de trabalhar, podemos ainda aplicar um grupo que abranja mais que uma coluna, utilizando o atributo `span="x"`. Assim, basta substituir o x pelo número de colunas que desejamos atribuir esse grupo.

Em seguida, vejamos então o exemplo:

```





```

Com isso, o resultado deste código será a seguinte tabela:

Nome	Idade	Profissão
Ted	22	Estudante
Junior	26	Escritor

Para finalizar esse tópico, recomendamos que você treine com diversas cores. Você pode utilizar a tabela de cores HTML.

7.5 Classes CSS (class)

Classes são seletores CSS. Utilizaremos seletores CSS para estilizar elementos no HTML, usando o atributo "class", por exemplo: <div class="blue"></div> em referência à uma classe CSS blue.

Primeiramente, CSS é uma linguagem que serve para estilizar sites. Ela descreve como deve ser a aparência dos elementos HTML em uma página web, e faz isso através de seletores.

Entenda seletores como palavras-chave com regras específicas que "selecionam" determinados elementos HTML em uma página, para poder atribuir uma aparência diferente a esses elementos.

Por exemplo, existem seletores CSS para selecionar tags HTML, tags com ids específicos, ex: <div id="footer"></div>, classes ex: <div class="footer"></div>, tags dentro de tags, entre outras regras mais específicas.

Veja o exemplo a seguir:

```
<style>
    div {
        color: red;
    }
</style>
<div>Teste</div>
```

O código CSS acima define um CSS selector "div", que vai selecionar todas as tags div na página e pintar o texto de vermelho.

Classes em CSS são simplesmente seletores. A sintaxe a ser utilizada é um ponto "." seguido de uma palavra-chave, e em seguida a definição dos estilos entre "{}".

Veja o exemplo a seguir:

```
<style>
    .bordered {
        border: 2px solid black;
    }
</style>

<div class="bordered"></div>
```

7.6 Inline

O CSS inline é usado para dar estilo a um elemento HTML específico. Para este estilo de CSS você somente vai precisar adicionar o atributo style para cada tag HTML, sem usar os seletores.

Este tipo de CSS não é realmente recomendado, já que cada tag HTML precisa ser estilizada de maneira individual.

Vamos dar uma olhada num exemplo.

Aqui, nós adicionamos um CSS inline para as tags <p> e <h1>:

```
<!DOCTYPE html>
<html>
<body style="background-color:black;">

<h1 style="color:white;padding:30px;">Hostinger Tutorials</h1>
<p style="color:white;">Something useful here.</p>

</body>
</html>
```

• **Vantagens do CSS Inline:**

1. Você pode inserir elementos CSS de maneira fácil e rápida numa página HTML. É por isso que esse método é útil para testar e pré-visualizar mudanças, assim como executar correções rápidas no seu site.
2. Você não precisa criar e fazer upload de um documento separado como no estilo externo.

• **Desvantagens do Inline CSS:**

1. Adicionar regras CSS para cada elemento HTML consome muito tempo e faz a sua estrutura HTML ficar bagunçada.
2. Estilizar múltiplos elementos pode afetar o tamanho da sua página e o tempo para download.

7.7 Block

Block (bloco) em uma página web é um elemento **HTML** que aparece em uma nova linha, Isto é, abaixo do elemento precedente em um modo de escrita horizontal e acima do elemento seguinte (comumente conhecido como um elemento de nível de bloco - block-level element). Por exemplo, **<p>** é por padrão um elemento em nível de bloco, enquanto **<a>** é um elemento inline (você pode colocar vários links próximos uns dos outros em seu HTML e eles ficarão na mesma linha como qualquer outro na saída renderizada).

Usando a propriedade **display** você pode alterar a forma como um elemento é exibido, como inline ou como block (bloco) (entre muitas outras opções); blocks (blocos) também estão sujeitos aos efeitos de esquemas de posicionamento e uso da propriedade **position**.

8.0 Prática

Crie uma lista ordenada utilizando as seguintes palavras:

Futebol, Basquete, Vôlei, Ginástica

Crie uma lista não ordenada utilizando as seguintes palavras:

Brasil, China, França, Suécia, Alemanha

Crie uma tabela e customize utilizando css contendo:

título Professores irá conter uma linha com nome dos professores e carga horária

Exemplo:

Professores:

Professor A	160h
Professor B	460h
Professor C	360h
Professor D	560h

9.0 O que são forms?

O **formulário HTML** é um formulário de preenchimento de dados ou que resulta em uma ação desejada utilizando a linguagem de marcação HTML. É formado por um ou mais widgets. Esses widgets são campos de textos, caixas de seleção, botões, radio buttons e **checkboxes** utilizando ferramentas do próprio HTML. Dessa forma, o usuário pode interagir com a página ao executar ações através desses formulários.

Esse recurso é muito utilizado para a criação de **formulários de contato**, formulários para captura de leads, e também para criação de sistemas, como por exemplo a criação de um modal de login.

Exemplo de uso:

```
<!-- Form que envia uma requisição GET -->
<form action="" method="get">
  <label for="GET-name">Name:</label>
  <input id="GET-name" type="text" name="name">
  <input type="submit" value="Save">
</form>

<!-- Simple form which will send a POST request -->
<form action="" method="post">
  <label for="POST-name">Name:</label>
  <input id="POST-name" type="text" name="name">
  <input type="submit" value="Save">
</form>

<!-- Form with fieldset, legend, and label -->
<form action="" method="post">
  <fieldset>
    <legend>Title</legend>
    <input type="radio" name="radio" id="radio"> <label for="radio">Click me</label>
  </fieldset>
</form>
```

9.1 Form, Validação

As validações de campos para formulário em **HTML5** é somente uma das maneiras de evitar o preenchimento robótico, e até mesmo confundir um cliente que não leu corretamente o que deveria ser preenchido em um campo específico.

Ocorre muito a confusão quando o campo é apropriado para telefone, e-mail e até mesmo o nome ou sobrenome em alguns casos. A validação permite reforçar o que às vezes já vem diretamente no código, como no caso do campo e-mail. Dessa forma estamos deixando abaixo algumas das mais importantes validações que você pode aplicar de forma simples e rápida em qualquer formulário em HTML5.

- **Apenas letras:**

```
<input type="text" required="required" name="text" pattern="[a-zA-Z\s]+$/>
```

- **Apenas números:**

```
<input type="text" required="required" name="numbers" pattern="[0-9]+\$/>
```

- **Data:**

```
<input type="date" required="required" maxlength="10" name="date" pattern="[0-9]{2}\V[0-9]{2}\V[0-9]{4}\$" min="2012-01-01" max="2014-02-18" />
```

- **Hora:**

```
<input type="time" required="required" maxlength="8" name="hour" pattern="[0-9]{2}:[0-9]{2} [0-9]{2}\$" />
```

- **Telefone:**

```
<input type="tel" required="required" maxlength="15" name="phone" pattern="[0-9]{2}-[0-9]{4,6}-[0-9]{3,4}\$" />
```

- **E-mail:**

```
<input type="email" required="required" class="input-text" name="email" pattern="[^@\w.]+\.[\w.]{2,4}\$" />
```

- **Desabilitar select automático de campos com opção para selecionar:**

value="" disabled selected

Você pode usar todo o input como no exemplo acima, ou apenas usar o pattern="" isso porque em alguns casos o seu input pode necessitar de uma classe específica para funcionar com o estilo certo.

Com isso você consegue inibir e dificultar o preenchimento incorreto dos seus campos no formulário. Sim, nós sabemos que somente essa ação não será 100% eficaz contra spam, existem inúmeros robôs forçando o preenchimento a todo instante, então assim como tudo na vida, precisamos evoluir constantemente os nossos métodos.

De toda forma você aplicando essas validações de campos para formulários em HTML5 você começa a desviar de spam com frequência em seus endereços.

9.2 Inputs

O elemento HTML <input> é usado para criar controles interativos para formulários baseados na web para receber dados do usuário. A semântica de um <input> varia consideravelmente dependendo do valor de seu atributo type.

O tipo de controle a ser exibido. O tipo padrão é text, se este atributo não for especificado.

Os valores possíveis são:

- **Button:** Um botão sem comportamento padrão.
- **Checkbox:** Uma caixa de marcação. Você deve usar o atributo **value** para definir o valor enviado por este item. Use o atributo checked para indicar se o item está selecionado por padrão. Você também pode usar o atributo **indeterminate** para indicar que a caixa de marcação está em um estado indeterminado (na maioria das plataformas, isso desenha uma linha horizontal cortando a caixa).
- **Color:** HTML5 Um controle para especificar cores. A interface de um seletor de cores não tem nenhuma funcionalidade obrigatória a não ser aceitar cores simples em texto.
- **Date:** HTML5 Um controle para inserir uma data (ano, mês e dia, sem horário).
- **Datetime:** HTML5 Um controle para inserir data e horário (hora, minuto, segundo e fração de segundo) baseado no fuso horário UTC.
- **Datetime-local:** HTML5 Um controle para inserir data e horário, sem fuso horário.

- **E-mail:** HTML5 Um campo para editar um endereço de e-mail. O valor do campo é validado para estar vazio ou ter um único endereço de e-mail válido antes de ser enviado. As pseudo classes CSS **:valid** e **:invalid** são aplicadas apropriadamente.
- **File:** Um controle que permite ao usuário selecionar um arquivo. Use o atributo **accept** para definir os tipos de arquivo que o controle pode selecionar.
- **Hidden:** Um controle que não é exibido mas cujo valor é enviado ao servidor.
- **Image:** Um botão gráfico para enviar o formulário. Você deve usar o atributo **src** para definir a fonte da imagem e o atributo **alt** para definir um texto alternativo. Você pode usar os atributos **height** e **width** para definir o tamanho da imagem em pixels.
- **Month:** HTML5 Um controle para inserir mês e ano, sem fuso horário.
- **Number:** HTML5 Um controle para inserir um número de ponto flutuante.
- **Password:** Um campo de texto com uma só linha cujo valor é obscurecido. Use o atributo **maxlength** para especificar o comprimento máximo do valor que pode ser inserido.
- **Radio:** Um botão de escolha. Você deve usar o atributo value para definir o valor a ser enviado por este item. Use o atributo **checked** para indicar se este item deve estar selecionado por padrão. Botões de escolha que têm o mesmo valor para o atributo name estão no mesmo "grupo de botões de escolha"; apenas um botão de escolha no grupo pode ser selecionado de cada vez.

- **Range:** HTML5 Um controle para inserir um número cujo valor exato não é importante. Este tipo de controle usa os seguintes valores padrão se os atributos correspondentes não forem especificados:
 - **min:** 0
 - **max:** 100
 - **value:** min + (max-min)/2, ou min se max for menos que min
 - **step:** 1
- **Reset:** Um botão que faz o conteúdo do formulário voltar a ter seus valores padrão.
- **Search:** HTML5 Um campo de texto com uma só linha para digitar termos de busca; quebras de linha são automaticamente removidas do valor entrado.
- **Submit:** Um botão que envia o formulário.
- **Tel:** HTML5 Um controle para inserir um número de telefone; quebras de linha são automaticamente removidas do valor entrado, mas nenhuma outra sintaxe é imposta. Você pode usar atributos como **pattern** e **maxlength** para restringir os valores inseridos no controle. As pseudo classes CSS **:valid** e **:invalid** são aplicadas apropriadamente.
- **Text:** Um campo de texto com uma só linha; quebras de linha são automaticamente removidas do valor entrado.
- **Time:** HTML5 Um controle para inserir um horário sem fuso horário.
- **Url:** HTML 5 Um campo para editar uma URL. O valor inserido é validado para ser vazio ou uma URL absoluta válida antes de ser enviado. Quebras de linha e espaços em branco antes e após o valor inserido são automaticamente removidos. Você pode usar atributos como **pattern** e **maxlength** para restringir os valores inseridos no controle. As pseudo classes CSS **:valid** e **:invalid** são aplicadas apropriadamente.
- **Week:** HTML5 Um controle para inserir uma data consistindo de ano da semana e número da semana sem fuso horário. Utilizando o **placeholder** é possível modificar o texto dentro do botão

9.3 Submit

Você usa esse tipo para adicionar um botão de envio aos formulários. Quando um usuário clica nele, ele envia automaticamente o formulário. Recebe um atributo value, que define o texto que aparece dentro do botão.

```
<input type="submit" value="Enter to Win" />
```

9.4 Action

O URI é um programa que processa as informações enviadas por meio do formulário. Esse valor pode ser substituído por um **formação atributo** em um **<button>** ou **<input>** elemento.

9.5 Get

O método GET significa recuperar qualquer informação (na forma de uma entidade) identificada pelo Request-URI. Se o Request-URI se referir a um processo de produção de dados, são os dados produzidos que devem ser retornados como a entidade na resposta e não o texto fonte do processo, a menos que esse texto seja a saída do processo.

A semântica do método GET muda para um "GET condicional" se a mensagem de solicitação incluir um campo de cabeçalho If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match ou If-Range. Um método GET condicional solicita que a entidade seja transferida apenas nas circunstâncias descritas pelo(s) campo(s) de cabeçalho condicional. O método GET condicional destina-se a reduzir o uso desnecessário da rede, permitindo que as entidades armazenadas em cache sejam atualizadas sem exigir várias solicitações ou transferir dados já mantidos pelo cliente.

A semântica do método GET muda para um "GET parcial" se a mensagem de solicitação incluir um campo de cabeçalho Range. Um GET parcial solicita que apenas parte da entidade seja transferida. O método GET parcial destina-se a reduzir o uso desnecessário da rede, permitindo que entidades recuperadas parcialmente sejam concluídas sem transferir dados já mantidos pelo cliente.

A resposta a uma solicitação GET pode ser armazenada em cache se e somente se atender aos requisitos para armazenamento em cache HTTP.

9.6 Post

O método **POST** é usado para solicitar que o servidor de origem aceite a entidade incluída na solicitação como um novo subordinado do recurso identificado pelo Request-URI na Request-Line. O POST foi projetado para permitir um método uniforme para cobrir as seguintes funções:

- Anotação dos recursos existentes;
- Postar uma mensagem em um quadro de avisos, grupo de notícias, lista de discussão, ou grupo similar de artigos;
- Fornecer um bloco de dados, como o resultado do envio de um formulário, a um processo de tratamento de dados;
- Estender um banco de dados por meio de uma operação de acréscimo.

A função real executada pelo método POST é determinada pelo servidor e geralmente depende do Request-URI. A entidade postada é subordinada a esse URI da mesma forma que um arquivo é subordinado a um diretório que o contém, um artigo de notícias é subordinado a um grupo de notícias ao qual é postado ou um registro é subordinado a um banco de dados.

A ação executada pelo método POST pode não resultar em um recurso que possa ser identificado por um URI. Nesse caso, 200 (OK) ou 204 (Sem conteúdo) é o status de resposta apropriado, dependendo de a resposta incluir ou não uma entidade que descreva o resultado. Se um recurso foi criado no servidor de origem, a resposta DEVE ser 201 (Criado) e conter uma entidade que descreve o status da solicitação e se refere ao novo recurso e um cabeçalho de localização

As respostas a esse método não podem ser armazenadas em cache, a menos que a resposta inclua os campos de cabeçalho Cache-Control ou Expires apropriados.

10 Prática

Crie um formulário de contato contendo os seguintes *inputs*:

Nome completo: deve ser do tipo texto

E-mail: deve ser do tipo e-mail

Município: deve ser do tipo texto

Mensagem: deve ser do tipo textarea



Digital College

ENSINO DE HABILIDADES DIGITAIS

digitalcollege.com.br