treinadev

Introdução a Web

CAMPUS CODE

Introdução a Web

Antes do amplo acesso de usuários domésticos à internet, o desenvolvimento de software estava fortemente ligado à criação de aplicações para desktop. Programas que eram instalados via CD, DVD ou disquete e cuja execução, geralmente, era independente da conexão com uma rede.

Com a expansão acelerada da internet no fim dos anos 1990 e começo dos anos 2000, a criação de programas voltados para esse novo formato cresceu até tornar-se o principal mercado para desenvolvedores. E assim segue até hoje. Mesmo agora, os populares aplicativos para celular utilizam aplicações Web como forma de armazenar e sincronizar dados.

Os principais fatores para a popularização desse novo modelo são a facilidade de distribuição de uma aplicação para todos usuários da internet e a possibilidade de mantê-los sempre atualizados com a versão mais recente do software, sem a necessidade de novas instalações.

Esse conteúdo vai apresentar a arquitetura por trás de aplicações Web com foco no protocolo HTTP, além de uma visão prática da construção de páginas HTML.

Arquitetura cliente-servidor

Em termos gerais, aplicações Web seguem um modelo de arquitetura chamado de cliente-servidor. Neste modelo, um servidor recebe e executa comandos solicitados por clientes através de uma rede. No caso da Web, as aplicações são executadas diretamente no navegador, sem a necessidade de instalação de um software específico, geralmente utilizando a internet como rede que liga clientes e servidores.

Na imagem, de um lado, temos clientes utilizando a aplicação, por exemplo, abrindo o Google Drive ou acessando o site da Campus Code. Do outro lado, temos servidores que armazenam e mantêm essas aplicações em execução a todo momento.

A arquitetura cliente-servidor pode ser usada de diferentes formas, seja na intranet de um hospital onde aplicações desktop acessam dados de um servidor ou em um restaurante onde *tablets* se comunicam via Wi-Fi com um desktop trocando informações sobre pedidos.

Mas, em nosso conteúdo, o foco é falar de aplicações Web que são acessadas por usuários através da internet. Então a partir de agora vamos focar nesse assunto. :)

Protocolo HTTP

Quando você digita um endereço no navegador, não sabe qual tecnologia foi usada naquele site/aplicação. Pode ser Ruby on Rails, PHP com Wordpress ou até várias linguagens ao mesmo tempo. Como é possível acessar essas aplicações no servidor sem ter nenhuma informação além de um simples endereço?

A navegação realizada na Web só é possível devido a algo que não damos muita importância ao navegar, as quatro letras antes do endereço: o HTTP ou *Hypertext Transfer Protocol*. Como o próprio nome diz, trata-se de um protocolo para transferência de hipertexto. Hipertexto é o formato mais comum para consumo de informações na Web e vamos falar mais sobre ele quando abordarmos HTML. Agora vamos nos aprofundar um pouco mais no HTTP.

Dica

O HTTPS é uma variação do HTTP em relação à segurança dos dados. No HTTPS, os sites possuem um certificado digital que permite a criptografia de dados transferidos com os clientes, garantindo assim maior segurança em transações financeiras e no uso de senhas, por exemplo.

Seguindo o modelo cliente-servidor, o HTTP possui uma requisição e uma resposta bem definidos. As aplicações Web devem ser capazes de receber essas requisições e devolver respostas dentro da especificação do protocolo.

Requisição

Quando acessamos o site da Campus Code e clicamos no link da Imersão Web, acessamos esta página: https://www.campuscode.com.br/imersao-web. Ao acessar essa página nosso navegador envia uma requisição HTTP para a internet.

Toda requisição HTTP é dividida em algumas partes, por agora vamos focar em duas: a URI e o método (ou verbo). Usando a requisição ao site da Campus Code, como exemplo temos:

- **URI**: www.campuscode.com.br/imersao-web
- **Método (verbo)**: GET

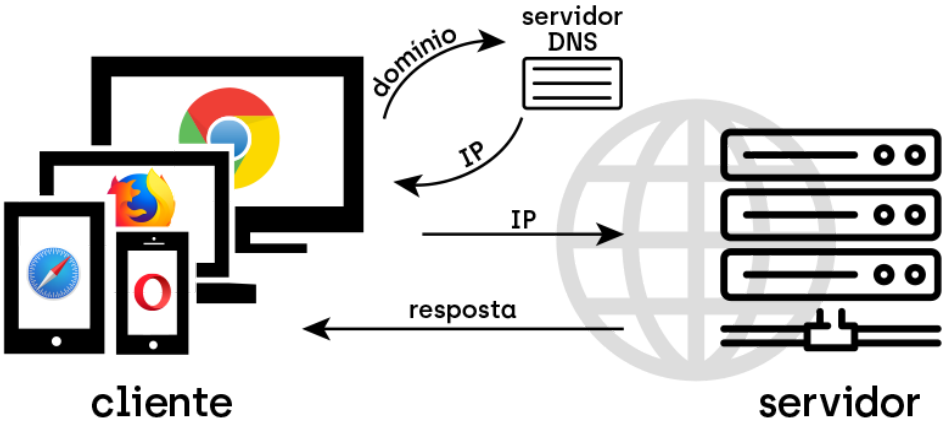
A URI (*Uniform Resource Identifier*) é o identificador do destino da nossa requisição. A forma mais comum de tratar a URI é separá-la em duas partes: o domínio e o caminho relativo.

Dica

A URL é um localizador completo que inclui, além do domínio e caminho relativo, o protocolo de acesso.
Por exemplo, https://www.campuscode.com.br/imersao-web é uma URL, assim como ftp://www.campuscode.com.br também é uma URL que utiliza o protocolo de transferência de arquivo, o FTP.

Ao digitar o domínio para navegar na internet, o que o navegador realmente quer saber é onde está o servidor que hospeda aquela aplicação. Cada servidor possui um identificador, o IP (*Internet Protocol*). A busca por esse endereço IP através do domínio é realizada pelo protocolo DNS, sigla de *Domain Name System*.

O DNS parte do navegador e passa pelo seu roteador até chegar a servidores espalhados pelo mundo que armazenam endereços dos domínios disponíveis na internet.



Quando vamos disponibilizar uma aplicação Web para o público, geralmente optamos por serviços de hospedagem como LocaWeb, Amazon ou Google Cloud. Os IPs retornados pelo DNS nos levam a esses servidores. Ao chegar lá, o domínio (campuscode.com.br) ainda é utilizado para localizar nossa aplicação, visto que a maioria dos servidores são compartilhados e hospedam diversos domínios ao mesmo tempo.

A outra parte da URI é o **caminho relativo**. Ele pode indicar um arquivo hospedado no servidor vinculado ao nosso domínio, mas também ser usado como parâmetro por uma aplicação Web para determinar a execução de algum código. Por exemplo:

- www.meusarquivos.com.br/arquivos/livro.pdf

Exemplo de URI com caminho relativo apontando diretamente para um arquivo hospedado no domínio meusarquivos.com.br.

- www.sitedenoticias.com/noticias/sp

Exemplo de URI com caminho relativo que serve como filtro para busca de notícias de São Paulo dentro da aplicação Web.

Dica

Quando acessamos um site ou aplicação sem informar um caminho relativo, o navegador automaticamente adiciona o caminho relativo ‘/’. É comum usarmos o termo ‘raíz’ para indicar o acesso a esse caminho, dado que ele representa a tela inicial da aplicação.

O outro elemento da requisição é o **método/verbo**. Os métodos HTTP mais comuns são: GET, POST e DELETE. Cada um deles indica o tipo de ação que deve ser tomada na URI da requisição HTTP. Um GET representa que queremos obter uma informação, já o POST indica a criação de novos dados na aplicação e o DELETE deve apagar dados.

Uma requisição HTTP difere de outra requisição pela junção de método e URI, o que permite termos duas requisições diferentes para a mesma URI, como no exemplo abaixo:

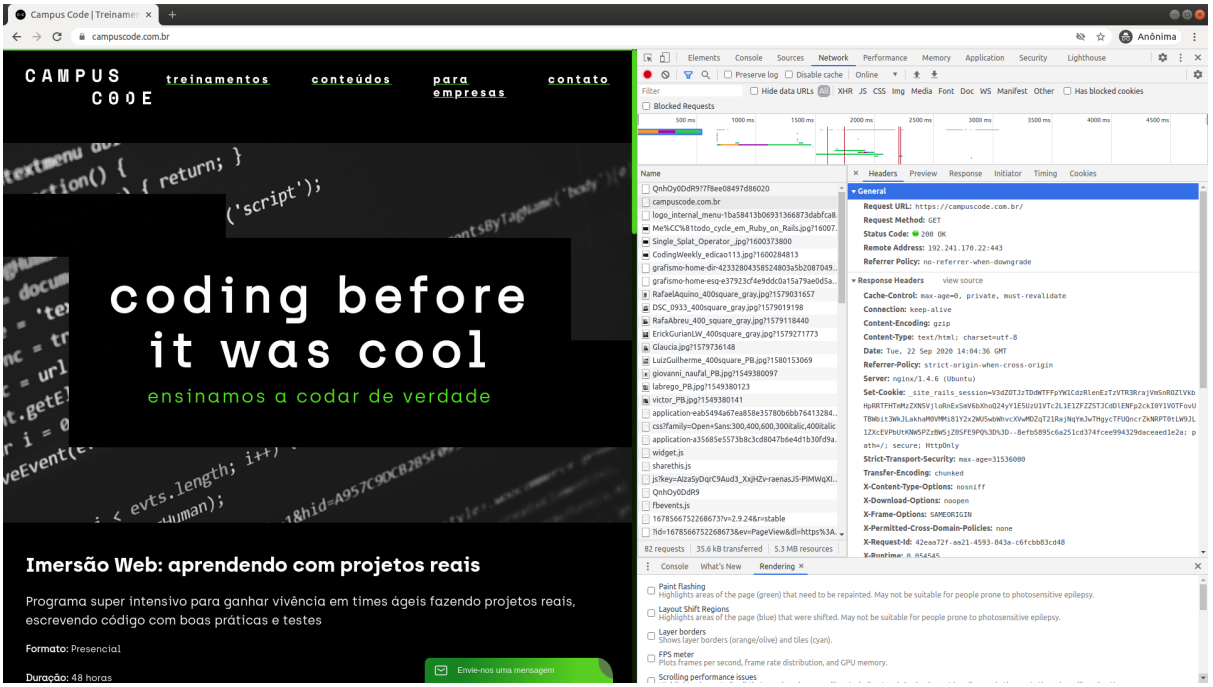
- GET sistema.com.br/alunos/32
- DELETE sistema.com.br/alunos/32

Dica

Existem outros métodos/verbos como PUT e PATCH, mas, a princípio, vamos usar apenas o GET e o POST. Se você quiser conhecer mais, aqui vai uma dica de leitura: Métodos de requisição HTTP.

Quer ver o que acontece na prática? Os navegadores possuem uma ferramenta que permite "olhar atrás dos bastidores" de uma página. Tanto no Firefox como no Chrome, você pode acessá-la clicando com o botão da direita do mouse em um local da página e indo até '*Inspect*'/Inspecionar.

Vamos fazer passo a passo. Abra um site qualquer no seu navegador (no exemplo, vamos usar o Google Chrome com o site da Campus Code), clique com o botão direito do mouse e, em seguida, em *'Inspect'*/Inspecionar. Na janela que se abriu, clique em *'Network'* e recarregue a página. Aparecerá um tipo de tabela com colunas como *'Name'*, *'Status'*, *'Type'* etc. Clique na segunda linha da tabela, logo abaixo de *'Name'* (que é a chamada) e, em seguida, em *'Headers'* caso já não esteja selecionado. Você verá a URI, o método chamado e o código do status, além das informações que o navegador disponibiliza, como a sua versão, a versão do Sistema Operacional, entre outras informações.



Resposta HTTP

Toda requisição recebe uma resposta que também é definida pelo HTTP. Aqui, vamos dividir a resposta em dois componentes principais: um código de status e um conteúdo.

O código é uma sequência de três dígitos que determina o resultado da execução da requisição. Os códigos são agrupados em categorias como sucesso, erro na requisição e erro no processamento, entre outros. O código **404** é um dos mais famosos e indica que a requisição foi feita para uma URI que não existe. Veja outros exemplos na tabela abaixo:

Código de Status	Resposta
200	Ok
204	Sem conteúdo
403	Não permitido
414	URI muito longa
500	Erro interno no servidor
503	Serviço indisponível

Dica

Se quiser conhecer todos os códigos de status, acesse: Códigos de status de respostas HTTP.

Junto ao código de status, toda resposta deve conter um conteúdo que é chamado de corpo da resposta. Esse corpo pode ter vários formatos, mas o mais comum é o HTML. HTML é uma linguagem de marcação que os navegadores de internet interpretam para gerar páginas que mesclam textos, imagens, áudios, vídeos e links para outras páginas.

Inicialmente todo conteúdo na internet se restringia a textos e existia apenas um punhado de mídias disponíveis para serem utilizadas nos sites. Apesar das tecnologias terem evoluído para incorporar outras mídias, como imagens, por exemplo, os sites ainda eram estáticos, ou seja, todo o conteúdo da página era carregado uma vez quando acessado pelo usuário. Atualmente as páginas são construídas de forma dinâmica e os conteúdos podem ser atualizados em tempo real enquanto o usuário navega pelo site. Além disso, o usuário pode ser requisitado a enviar dados por meio de formulários, fazer compras, assistir filmes etc. O aumento da sofisticação das interações entre usuário e site, tornou-se possível graças ao surgimento de aplicações Web que permitem realizar pelo navegador tarefas que antes eram realizadas somente via softwares instalados nos computadores.

HTML, CSS e JavaScript

HTML (*Hypertext Markup Language*) é uma linguagem que usa um sistema de marcadores (também chamados de *tags*) que representam um conteúdo específico para construir documentos na *World Wide Web* (WWW). Com esses marcadores, o HTML declara o que será apresentado, mas não como ele é visualizado. A representação visual é definida pelo CSS (*Cascading Style Sheets*). Além dessas duas linguagens, temos ainda o JavaScript. Ele permite enviarmos código integrado ao HTML e CSS, mas que será executado diretamente no navegador do usuário. Sabe quando você navega por um site que a interface muda de acordo com eventos ou interações suas? Provavelmente temos JavaScript sendo executado por trás.

Mas, por que aprender HTML, CSS e JavaScript? Se você pretende trabalhar com tecnologias Web, boa parte dos projetos envolverá o uso de um navegador. Seja o sistema de um banco ou uma rede social, o meio mais comum de utilização é a Web, por isso é interessante ter uma boa base de conhecimento sobre essas linguagens. O objetivo desse conteúdo é apresentar a base de HTML e CSS. O caso do JavaScript merece ser tratado à parte visto que é um assunto muito amplo e a tecnologia está em constante expansão. Além disso, foge do escopo pretendido para nossos treinamentos.

Elementos e Marcadores

Como foi explicado anteriormente, quando uma pessoa acessa um determinado endereço na internet, o servidor envia uma resposta e o navegador recebe um documento HTML que descreve a estrutura da página Web que será exibida. Os elementos HTML são as unidades que constituem as páginas Web.

Esses elementos são basicamente marcadores que atribuem significado semântico e envolvem o conteúdo da página. Um elemento normalmente consiste em um marcador de abertura (`<marcador>`) e um marcador de fechamento (`</marcador>`), que contém o nome do marcador envolvido por `<>`. Entre as *tags* entra o conteúdo: `<marcador>` … conteúdo … `</marcador>`.

No exemplo abaixo temos um conteúdo com marcador que indica tratar-se de um conteúdo associado ao rodapé da página:

```
<footer>
    Criado por Campus Code
</footer>
```

Dica

Alguns elementos HTML não possuem marcadores de fechamento ou conteúdo, por exemplo: `<img>`, `<meta>`, `<link>` e `<input>`.

Cada elemento pode conter também atributos que possuem um nome e um valor. No exemplo abaixo o marcador `<img>` tem um atributo `src`, que indica o endereço da imagem que deve ser exibida na página:

```
<img src="http://exemplo.com/imagem.jpg">
```

Cada marcador tem atributos diferentes de acordo com seu objetivo, mas alguns atributos são comuns a todos marcadores HTML sendo o mais importante o `id`. Este atributo serve para identificarmos unicamente um elemento em toda página. Atualmente temos várias outras tecnologias que interagem com páginas HTML e o uso de identificadores facilita essa integração.

```
<p id="descricao">Descrição de um workshop da Campus Code</p>
<p id="autor">José da Silva</p>
```

Uma página HTML completa pode conter centenas de elementos. O navegador lê, interpreta esses elementos e então exibe na tela o conteúdo da página Web.

O HTML está em constante evolução, por isso recomendamos que você consulte o site de referência oficial W3Schools. No entanto, vamos apresentar resumidamente alguns dos principais marcadores e seus atributos para você continuar os estudos.

Estrutura do documento HTML

Vamos começar pela estrutura básica de um documento HTML (indentação adicional para deixar mais clara hierarquia das *tags*):

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

    <h1>Título</h1>
    <p>Esse é um parágrafo.
          <img src='http://campuscode.com.br/logo.png' alt="Logo da
Campus Code">
    </p>

</body>
</html>
```

Os documentos HTML devem sempre começar com a declaração do seu tipo: `<!DOCTYPE html>`. O documento em si, começa com o marcador `<html>` e termina no `</html>`. Dentro da *tag* HTML existem sempre dois marcadores: `<head>` e `<body>`. Dentro da *tag* `<head>` são colocados metadados da página. O conteúdo da página que fica visível deve ficar entre os marcadores `<body>` e `</body>`.

Comentários

O `<!-- -->` comentário em HTML pode ser usado de duas formas: comentar uma única linha ou para bloco de linhas.

```
<!--
    <h1>Isto é um comentário</h1>
    <p>Ao ser um comentário não aparecerá nada na página</p>
    <p>Nem as TAGS HTML </p>
-->
```

Metadados

O `<head>` é o local para inserção dos marcadores de metadados, que são basicamente "dados sobre dados". Eles contém informações sobre o documento HTML que não serão exibidas (como título, descrição, palavras-chaves e autor, entre outras), mas que serão utilizadas pelo navegador ou serviços de internet para identificar e caracterizar a sua página ou aplicação Web. Vamos ver em detalhes algumas das principais *tags* de metadados.

O `<title>` define o título na barra de navegação do navegador e nos seus favoritos e exibe o título da página nos resultados em serviços de busca. Por isso, é um marcador obrigatório. Vale ressaltar que só pode haver um elemento `<title>` por documento.

```
<head>
    <title>Campus Code</title>
</head>
```

O `<style>` é o local para definir os estilos de um documento HTML, ou seja, ele determina como os elementos da página devem ser visualmente apresentados. O código desses estilos utiliza a formatação CSS. Vamos apresentar mais detalhes sobre CSS na continuação desse texto, mas apresentamos abaixo um exemplo que define que todos os cabeçalhos com a *tag* `<h1>` devem aparecer com a fonte na cor vermelha e todos os parágrafos da página devem ser exibidos com a fonte em azul.

```
<head>
    <style>
        h1 {color:red;}
        p {color:blue;}
    </style>
</head>
```

Você também pode conectar o documento a um arquivo externo de estilos utilizando o marcador `<link>`. Veremos mais sobre estilos quando falarmos sobre CSS. Essa *tag* pode ser utilizada para diversos tipos de arquivo, embora seja utilizada principalmente para importar os estilos de arquivos CSS. Esse elemento pode aparecer múltiplas vezes num mesmo documento dentro do `<head>`.

```
<head>
    <link rel="stylesheet" type="text/css" href="theme.css">
</head>
```

O marcador `<script>` é utilizado para definir códigos JavaScript que serão rodados no lado do cliente. O `<script>` pode conter os códigos declarados ou apontar para arquivos externos que são carregados pelo atributo `src`.

```
<script>
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

Vale ressaltar que esse marcador não precisa estar no `<head>` necessariamente. Ele também pode ser definido em outros locais do documento HTML, sendo importante avaliar caso a caso qual o melhor local para que seja inserido.

Finalmente, o marcador `<meta>` é utilizado para especificar uma série de informações, como o conjunto de caracteres utilizados no documento (com o atributo `charset`) , a descrição e as palavras-chave da página, autor, entre outros metadados. Essas informações são muito importantes para que os navegadores saibam como exibir o conteúdo corretamente e para que serviços na internet tenham acesso a dados relevantes, como as palavras-chave relacionadas à sua página para ferramentas de busca.

A seguir, apresentamos um exemplo de como montar uma estrutura de metadados no seu documento:

```
<head>
    <meta charset="UTF-8">
    <meta name="description" content="Free Web tutorials">
    <meta name="keywords" content="HTML,CSS,XML,JavaScript">
    <meta name="author" content="John Doe">
</head>
```

Os marcadores `<meta>` sempre devem ser definidos dentro do `<head>` e seu conteúdo nunca será exibido em tela, apesar de estarem disponíveis como informações.

Corpo

O marcador `<body>` envolve todo conteúdo que vai ser exibido no navegador. Dentro desse elemento encontramos outros elementos que contém textos, imagens, formulários etc. Esses elementos podem inclusive ser aninhados, como no nosso exemplo de estrutura HTML onde a imagem está dentro de um parágrafo.

```
<p>Esse é um parágrafo.
    <img src='https://campuscode.com.br/logo.png' alt="Logo da Campus
Code">
</p>
```

Nos próximos tópicos vamos apresentar os elementos mais usados dentro do corpo de uma página.

Cabeçalho e parágrafo

O marcador `<h1>` determina o conteúdo do cabeçalho. Você pode usar variações desses marcadores de `<h1>` a `<h6>` para cabeçalhos de diferentes prioridades, sendo que o `<h1>` é o de maior importância, diminuindo até o menor `<h6>`. O marcador `<p>` determina um parágrafo de texto.

```
<h1>Cabeçalho H1</h1>
<p>Esse é um parágrafo de texto.</p>
<h2>Cabeçalho H2</h2>
<p>Esse é outro parágrafo de texto.</p>
```

O marcador `<br>` insere uma quebra de linha. Não é necessária *tag* de fechamento.

Imagem

Para inserir imagens na sua página você deve usar o marcador `<img>`.

```
<img src="campuscode.png" alt="Campus Code" width="104" height="142">
```

Note que, como foi mencionado anteriormente, o `<img>` não tem marcador de fechamento. Ele possui atributos disponíveis para sua manipulação como o `src` (define o arquivo de origem), `alt` (texto alternativo), `width` (largura) e `heigth` (altura). Você pode consultar o site da W3Schools para ver todos os atributos disponíveis para esse marcador.

Listas

As formas mais comuns de montar listas são as listas ordenadas e as não ordenadas. Listas ordenadas utilizam numeração sequencial para organizar os itens da lista, enquanto listas não ordenadas utilizam símbolos como • (bullet).

No HTML, listas ordenadas começam com `<ol>` e listas não ordenadas com `<ul>`. Cada item da lista deve entrar com o marcador `<li>`.

```
<ol>
    <li>Item da lista ordenada</li>
    <li>Item da lista ordenada</li>
    <li>Item da lista ordenada</li>
</ol>

<ul>
    <li>Item da lista não ordenada</li>
    <li>Item da lista não ordenada</li>
    <li>Item da lista não ordenada</li>
</ul>
```

Tabelas

Tabelas em documentos HTML são definidas por meio da *tag* `<table>`. Cada linha deve ser definida com o marcador `<tr>` e cada célula numa linha é determinada com o marcador `<td>`. As células do cabeçalho da tabela são definidas com a *tag* `<th>`. A seguir, apresentamos um exemplo de estrutura de uma tabela no documento HTML:

```
<table>
    <tr>
        <th>Nome</th>
        <th>Sobrenome</th>
        <th>Idade</th>
    </tr>
    <tr>
        <td>João</td>
        <td>Silva</td>
        <td>50</td>
    </tr>
    <tr>
        <td>Evandro</td>
        <td>Almeida</td>
        <td>94</td>
    </tr>
</table>
```

Tabelas possuem uma série de atributos que permitem configurá-las da forma que for necessário. Você pode estudar mais sobre tabelas neste artigo da W3Schools.

Interações com as páginas

As duas formas convencionais que permitem a interação de usuários com o conteúdo de uma página são os links e os formulários. Enquanto os links têm como propósito nos levar a outra página, os formulários permitem a inserção de dados que são submetidos para nossa aplicação. Nos dois cenários, a interação do usuário faz com que o navegador inicie uma nova requisição HTTP.

Links

Se você quiser adicionar links na sua página, deve utilizar o marcador `<a>` junto com o atributo `href`. Este último especifica o destino do link. Acesse o artigo da W3Schools para saber mais sobre links e seus atributos.

```
<a href="https://www.campuscode.com.br">Texto do link</a>
```

Um link suporta, como conteúdo a ser exibido, desde um texto simples sem necessidade de marcadores adicionais, até blocos inteiros de elementos HTML, como no exemplo a seguir:

```
<a href="https://www.campuscode.com.br">
    <h1>Campus Code</h1>
    <img src="https://campuscode.com.br/logo.png" alt="Logotipo da
Campus Code">
</a>
```

Ao acionar um link, a nova requisição HTTP gerada pelo navegador é, por padrão, um GET para o endereço indicado no atributo `href`.

Formulários

Um formulário é o exemplo mais complexo de elemento HTML que vamos abordar neste conteúdo. Todo formulário possui campos que permitem a entrada de dados pelo usuário e um botão que confirma o envio dos dados. Ao enviar esses dados, o navegador dispara uma nova requisição HTTP. A URI e o método HTTP usados nesta requisição são definidos como atributos no formulário.

Para criar formulários, o HTML possui o marcador `<form>` que encapsula os campos de inserção de dados e os botões de submissão. Os atributos `action` e `method` são muito importantes, pois definem qual requisição HTTP o navegador vai usar para enviar os dados do formulário. No atributo `action` devemos informar o caminho relativo ou URI completa e no atributo `method` qual método HTTP será utilizado. Se queremos cadastrar uma nova pessoa em uma aplicação com uma requisição POST para o caminho 'exemplo.com/novo_cadastro', devemos criar então o formulário como abaixo:

```
<form action="/novo_cadastro" method="post">
</form>
```

O código acima não produz nenhuma representação visual do formulário ao ser visualizado no navegador. Os formulários agrupam diversos elementos do tipo `label` e `input` que têm essa responsabilidade.

No exemplo a seguir, é gerado um formulário com campos que pedem o nome e a idade do usuário, além de um botão para enviar os dados inseridos.

```
<form action="/novo_cadastro" method="post">
    <label>Nome:</label> <input type="text" name="nome"><br>
    <label>Idade:</label> <input type="text" name="idade"><br>
    <input type="submit" value="Enviar">
</form>
```

Os marcadores `<input>` definem os tipos de elementos do formulário por meio do atributo `type`. Os campos nome e idade são do tipo `text`, onde o usuário escreve informações como texto, enquanto o botão para enviar os dados é do tipo `submit`. O HTML disponibiliza uma série de elementos para personalizar o formulário de acordo com suas

necessidades. Você pode utilizar áreas de texto e campos seletores de opções, entre outros. Recomendamos que você acesse os artigos da W3Schools sobre formulários e faça alguns experimentos para se habituar com as opções, a estrutura e conhecer algumas boas práticas para a construção de formulários.

HTML5 e Semântica

Nas primeiras versões do HTML e no começo do acesso em massa de páginas Web ainda era comum o uso de marcadores HTML que, além de representar o contexto de determinado conteúdo, determinavam a forma como o conteúdo seria exibido. Marcadores como `<center>`, `<b>`, `<font>`, `<marquee>` e muitos outros foram descontinuados a cada atualização do HTML e seu uso hoje não é mais recomendado.

Em algum momento dessa breve história, até mesmo tabelas eram usadas para organizar o conteúdo na tela. Na época, parecia uma boa ideia separar cabeçalhos, menus, conteúdo e rodapé como uma grande tabela, mas o surgimento de cada vez mais dispositivos com diferentes resoluções e formatos de tela derrubou rapidamente essa forma de construção de páginas.

Dica

O movimento Tableless surgiu e ganhou muita força nos anos 2000, promovendo o uso consciente de HTML e CSS para evitar o uso de tabelas na construção de páginas. O site https://tableless.com.br surgiu nessa época e é uma comunidade ativa até hoje para promover discussões sobre boas práticas de Web

Com o tempo, o marcador `<div>` (abreviação de *divider*) ganhou espaço como a principal forma de organizar conteúdo nas páginas, que ficavam cada vez maiores. A função desse marcador é realmente organizar conteúdos em blocos. Veja um exemplo de uso no trecho de HTML abaixo:

```
<div id="menu">
    <a href="/noticias">Notícias</a>
    <a href="/contato">Contato</a>
</div>
<div id="conteudo">
    <h1>Notícias da Semana</h1>
    <a href="http://outrapagina.com.br/>Link no meio do conteúdo</a>
</div>
<div id="rodape">
    <img src='https://campuscode.com.br/logo.png' alt="Logo da Campus
Code">
</div>
```

Parece tudo certo, né? Temos três `<div>` organizando menu, conteúdo e rodapé. Até certo ponto da trajetória das aplicações e páginas Web, estava mesmo correto, mas dois fatores levaram a mais uma evolução na construção das páginas:

- a criação de páginas com maior acessibilidade, com apoio de ferramentas como leitores de telas, por exemplo;
- o surgimento de ferramentas como Google, que analisam as páginas para realizar buscas relevando a qualidade do conteúdo.

O uso de elementos `<div>` já não era suficiente, pois não havia uma padronização. Ao criar uma página, cada pessoa poderia usar esse elemento com diferentes propósitos e usando identificadores diferentes, sem contar o fato dos identificadores muitas vezes serem definidos na língua nativa da pessoa. Nosso exemplo acima reflete esse cenário: usamos menu, conteúdo e rodapé como identificadores, mas dificilmente o Google ou um leitor de telas entenderia essa nomenclatura. Esse problema foi resolvido com o lançamento do HTML5.

O HTML5 trouxe novos marcadores responsáveis, exclusivamente, pela organização padronizada do conteúdo de uma página. Veja exemplos desses marcadores:

Marcador	**Propósito**
<nav></nav>	Agrupa links relacionados à navegação da página, como, por exemplo, o menu de uma página ou aplicação.
<article></article>	Envolve um conjunto de elementos, compondo um conteúdo que pode ser até isolado do restante da página e ainda deve fazer sentido.
<section></section>	Seções da página ou de um <article>.
<main></main>	Encapsula o conteúdo principal da página.
<header></header>	Cabeçalho de uma página ou de uma seção de uma página.
<footer></footer>	Rodapé de uma página ou de uma seção de uma página.

Dica

Na W3Schools existe uma página que trata somente de elementos semânticos do HTML5: HTML5 Semantic Elements.

A semântica aplicada pelos marcadores não influencia em nada o visual das páginas. Toda composição visual de uma página HTML deve ser determinada pelo seu estilo CSS.

CSS

O CSS provê estilos aos elementos HTML de uma página. Basicamente, descreve a forma como os elementos devem ser exibidos.

Ele pode ser aplicado de três maneiras:

- *Inline*, indica as alterações dentro dos marcadores HTML. É desencorajada a utilização desse método;
- Interno, utiliza o marcador `<style>` para definir dentro da própria página os estilos do CSS;
- Externo, dentro do marcador `<link>` é referenciado um arquivo externo com os estilos que devem ser aplicados ao documento HTML. Este é o método mais recomendado.

CSS *Inline*

O CSS *inline* aplica um estilo num só elemento HTML, ou seja, caso você queira alterar a cor de todos os cabeçalhos `<h3>`, por exemplo, será necessário modificar cada ocorrência da *tag* no documento inteiro. O exemplo a seguir mostra a aplicação do CSS inline em um elemento `<h3>` para que ele seja vermelho.

```
<h3 style="color: red;">Esse cabeçalho é vermelho</h3>
```

CSS interno

O CSS interno é usado para definir os estilos de um único documento HTML. Ele deve ser escrito no `<head>` do HTML dentro de um marcador `<style>`. O exemplo abaixo define a cor de fundo do `<body>` para verde claro, os cabeçalhos `<h1>` para cor azul e os parágrafos para terem texto em cor branca com fundo verde:

```
<head>
    <style>
        body {background-color: lightgreen;}
        h1   {color: blue;}
        p    {color: white; background-color: green;}
    </style>
</head>
```

A desvantagem deste formato é que numa aplicação ou site com diversas páginas HTML teremos que repetir o estilo em cada uma das páginas para manter a mesma identidade visual.

CSS externo

Para acessar os estilos externos, é necessário criar um arquivo com extensão .css e apontá-lo no `<head>` do seu documento com o marcador `<link>`:

```
<head>
    <link rel="stylesheet" type="text/css" href="stylesheet.css">
</head>
```

No lugar de usar um arquivo personalizado, você também pode utilizar arquivos externos fornecidos por um site como o Bootstrap ou Materialize:

```
<head>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
</head>
```

A vantagem de usar um arquivo de estilos externo é a facilidade com que você pode atualizar todo o seu site. Como todas as informações são armazenadas em um só documento, você só precisa modificar os estilo uma única vez.

Um arquivo de estilos externo deve parecer com o exemplo abaixo:

```
h1 {
  color: blue;
  font-family:verdana;
  font-size: 300%;
}
h2 {
  color: blue;
  font-family:verdana;
  font-size: 200%;
}
p {
  color: red;
  font-family:courier;
}
```

Propriedades CSS

Nos exemplos apresentados anteriormente, já mostramos como manipular os elementos HTML alterando os valores das propriedades, como `color`, por exemplo. O CSS possui

muitas propriedades que podem ser utilizadas para manipular a forma como o conteúdo HTML é exibido na tela. Vamos falar brevemente sobre algumas delas.

Para modificar os textos, temos as propriedades `color` (para cor), `font-family` (para o tipo de fonte) e `font-size` (para o tamanho da fonte).

```
p  {
    color: red;
    font-family: courier;
    font-size: 160%;
}
```

As propriedades `border`, `padding` e `margin`, definem respectivamente a borda, o espaço entre o texto e a borda, e a margem nos elementos HTML.

```
p {
    border: 1px solid powderblue;
    padding: 30px;
    margin: 50px;
}
```

Você pode estudar mais sobre CSS e praticar no tutorial da W3Schools.

Atributos id e class

Como já foi explicado anteriormente, o atributo `id` é utilizado para identificar um elemento específico no documento HTML. Com ajuda dele, o CSS pode ser utilizado para atribuir a esse elemento um estilo específico. O elemento HTML identificado com `id="paragrafo01"` no documento HTML:

```
<p id="paragrafo01">Eu sou um parágrafo diferente.</p>
```

O CSS define o estilo desse elemento específico, identificando o `id` com `#`:

```
#paragrafo01 {
    color: red;
}
```

O atributo `id` também pode ser utilizado com JavaScript para executar códigos e manipular elementos únicos.

Outra maneira de aplicar estilos no documento HTML é com o atributo `class`, utilizado para definir o estilo de todos os elementos com o mesmo nome do atributo `class`. No exemplo a seguir temos a `class="citacao"` aplicada em dois elementos `<p>`.

```
<p>Eu sou um texto padrão.</p>
<p class="citacao">"Eu sou um texto de citação."</p>
<p class="citacao">"Eu sou um texto de citação."</p>
<p>Eu sou um texto padrão.</p>
```

No CSS, o estilo da `class` deve ser indicado com um `.` antes do nome:

```
p {
  margin: 15px;
  padding: 15px;
}
.citacao {
  background-color: black;
  color: white;
  margin: 20px;
  padding: 20px;
  font-size: 160%;
}
```

A grande diferença entre `id` e `class` é que o primeiro deve estar aplicado em um único elemento, enquanto `class` pode ser utilizado em múltiplos elementos que serão exibidos com o mesmo estilo.

Chamamos de seletores CSS a simbologia para aplicar um estilo em todos elementos de um marcador ou somente elementos de determinado `id` ou `class` como visto acima. Mas existem diversos outros seletores. Para consultar uma versão sempre atualizada dos seletores recomendamos o artigo da W3Schools: CSS Selector Reference.

Da mesma forma, existem dezenas de propriedades além das apresentadas aqui. Mais uma vez recomendamos consultar e pesquisar mais no site da W3Schools: CSS Properties.

Indo além

Chegamos ao fim do nosso conteúdo de introdução a tecnologias Web. Acreditamos que aprender e saber utilizar as bases de HTTP e HTML permite que você aprenda qualquer tecnologia que surgiu nos últimos anos para criar aplicações mais robustas e complexas.

Esse material está em constante evolução e sua opinião é muito importante. Se tiver sugestões ou dúvidas que gostaria de nos enviar, entre em contato pelo nosso e-mail: treinadev@campuscode.com.br.

Versão	Data de atualização
1.0.4	20/06/2022

treinadev

é um programa gratuito de
formação de devs da Campus Code

Apoio:

VINDI

REBASE

PORTAL solar

smart fit

BRAINN

CAMPUS CODE