

Desenvolvimento Web



Autor: Gabriel Felippe

Contato: gabrielfelippe@outlook.com

Website: <http://akiradev.netlify.app>

Introdução

- O Desenvolvimento Web é o trabalho envolvido no desenvolvimento de websites para a Internet (World Wide Web) ou uma intranet (rede privada).
- Ele pode variar desde o desenvolvimento de uma única página estática simples até complexos aplicativos de Internet, negócios eletrônicos e serviços de rede social.

World Wide Web (WWW)

- A World Wide Web (WWW), conhecida como Web, é um sistema de informação onde documentos e outros recursos da web são identificados por Uniform Resource Locators (URLs, como <https://example.com>).
- Os recursos podem ser interligados por hipertexto e são acessíveis pela Internet.
- Eles são transferidos por meio do Protocolo de Transferência de Hipertexto (HTTP), podem ser acessados pelos usuários por um software denominado web browser e são publicados por um software conhecido como web server.

Breve Histórico

- O cientista inglês Sir Timothy Berners-Lee inventou a World Wide Web em 1989.
- Ele escreveu o primeiro web browser em 1990, enquanto trabalhava no CERN perto de Genebra, Suíça.
- A Web começou a entrar no uso diário em 1993, quando sites de uso geral começaram a se tornar disponíveis.
- Ela foi fundamental para o desenvolvimento da Era da Informação e é a principal ferramenta que bilhões de pessoas usam para interagir na Internet.

World Wide Web Consortium

- O World Wide Web Consortium (W3C) é a principal organização de padrões internacionais para a World Wide Web.
- Foi fundado no Massachusetts Institute of Technology Laboratory for Computer Science (MIT / LCS) com o apoio da Comissão Europeia, a Defense Advanced Research Projects Agency (DARPA), que foi pioneira na ARPANET, uma das predecessoras da Internet.



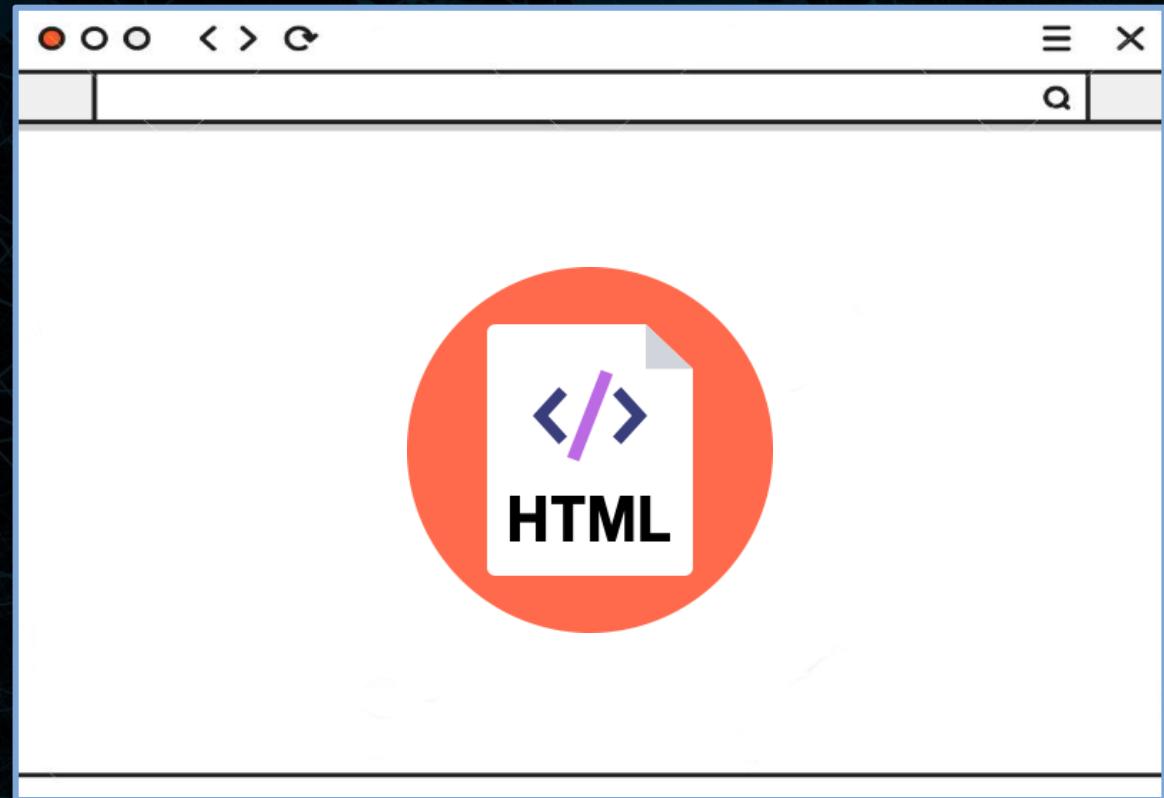
Páginas Web

- Web Browsers (Navegadores) são aplicações que podem apresentar páginas web.
- Exemplos: Chrome, Firefox, Safari, Internet Explorer, Brave, Tor, Microsoft Edge, Opera.



Páginas Web

- As páginas da web são escritas em uma linguagem de marcação chamada HTML, de modo que os browsers exibem uma página da web lendo e interpretando seu HTML.



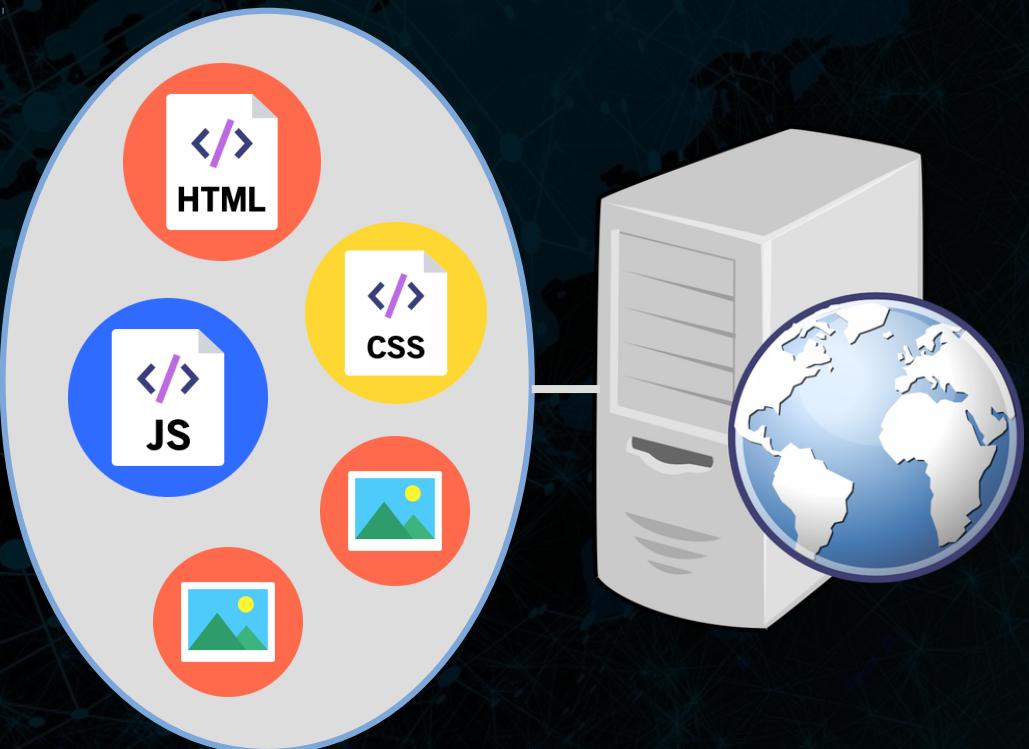
Páginas Web

O arquivo HTML pode se vincular a outros recursos, como imagens, vídeos, bem como arquivos JavaScript e CSS (folha de estilo), que o browser também carrega.

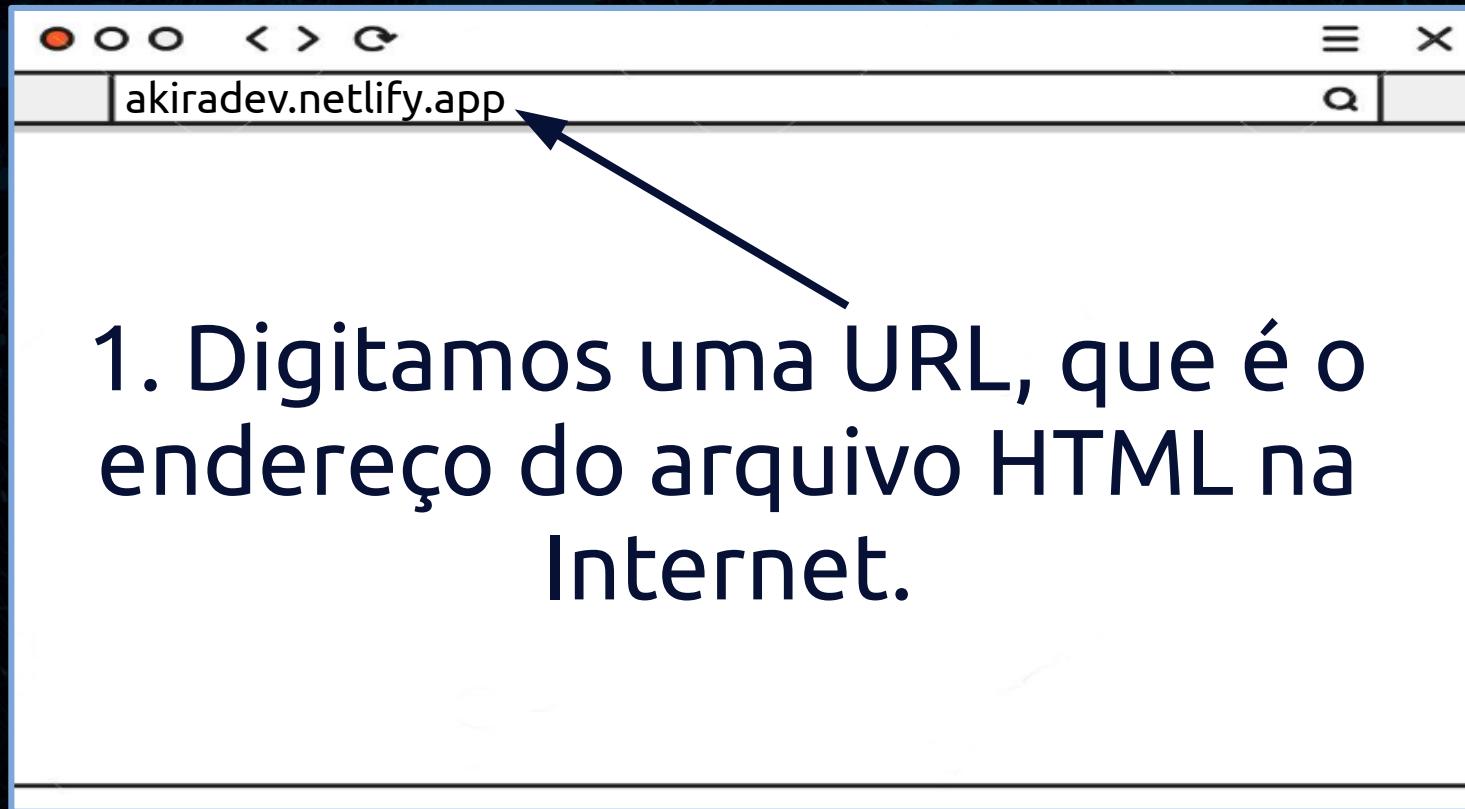


Páginas Web

- Um servidor web é um programa executado em um computador que entrega páginas da web em resposta a solicitações.
- Ele armazena ou gera a página da web retornada.



Funcionamento



1. Digitamos uma URL, que é o endereço do arquivo HTML na Internet.

Funcionamento



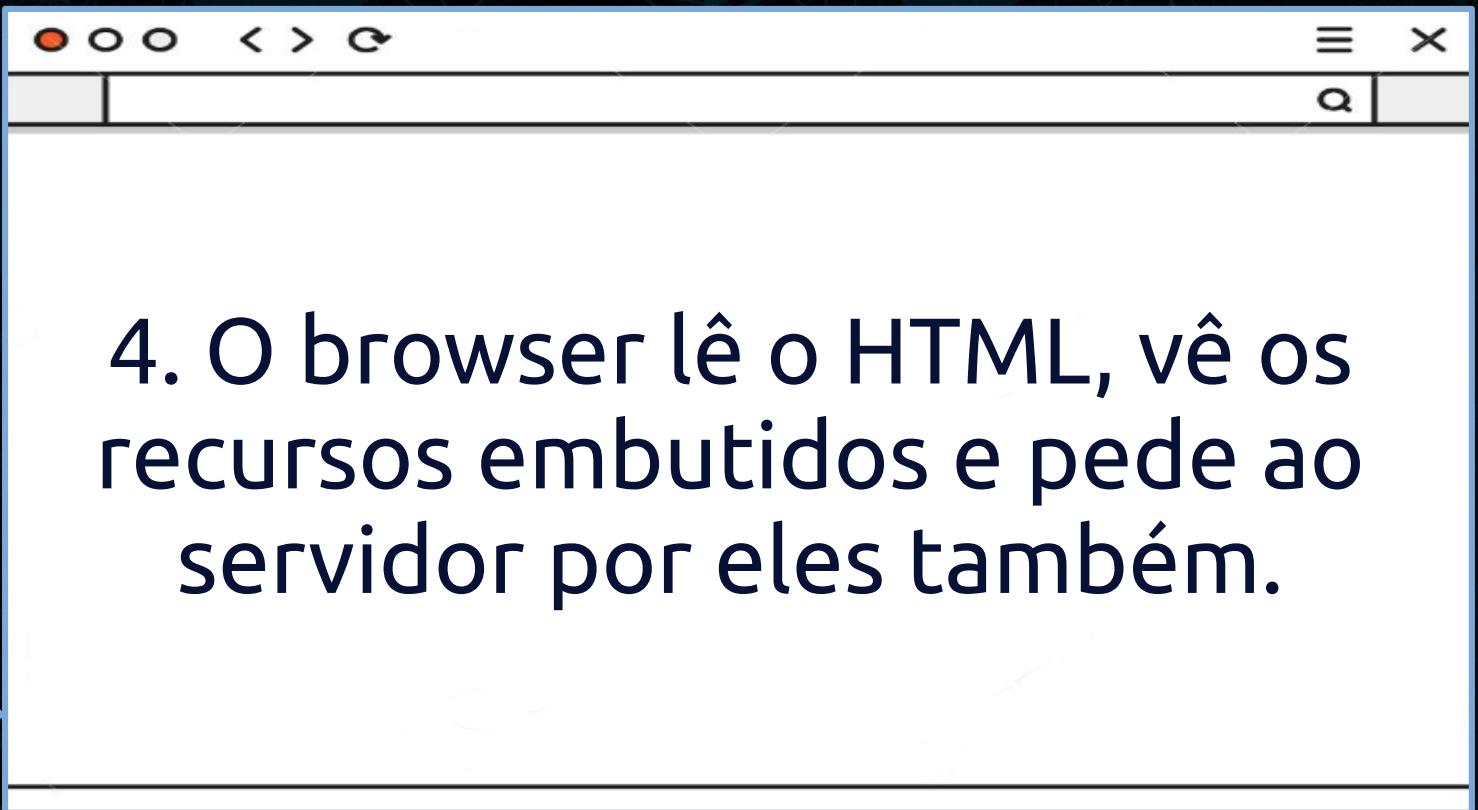
2. O browser pede ao servidor da web (que hospeda o documento) para enviar o documento.

Funcionamento



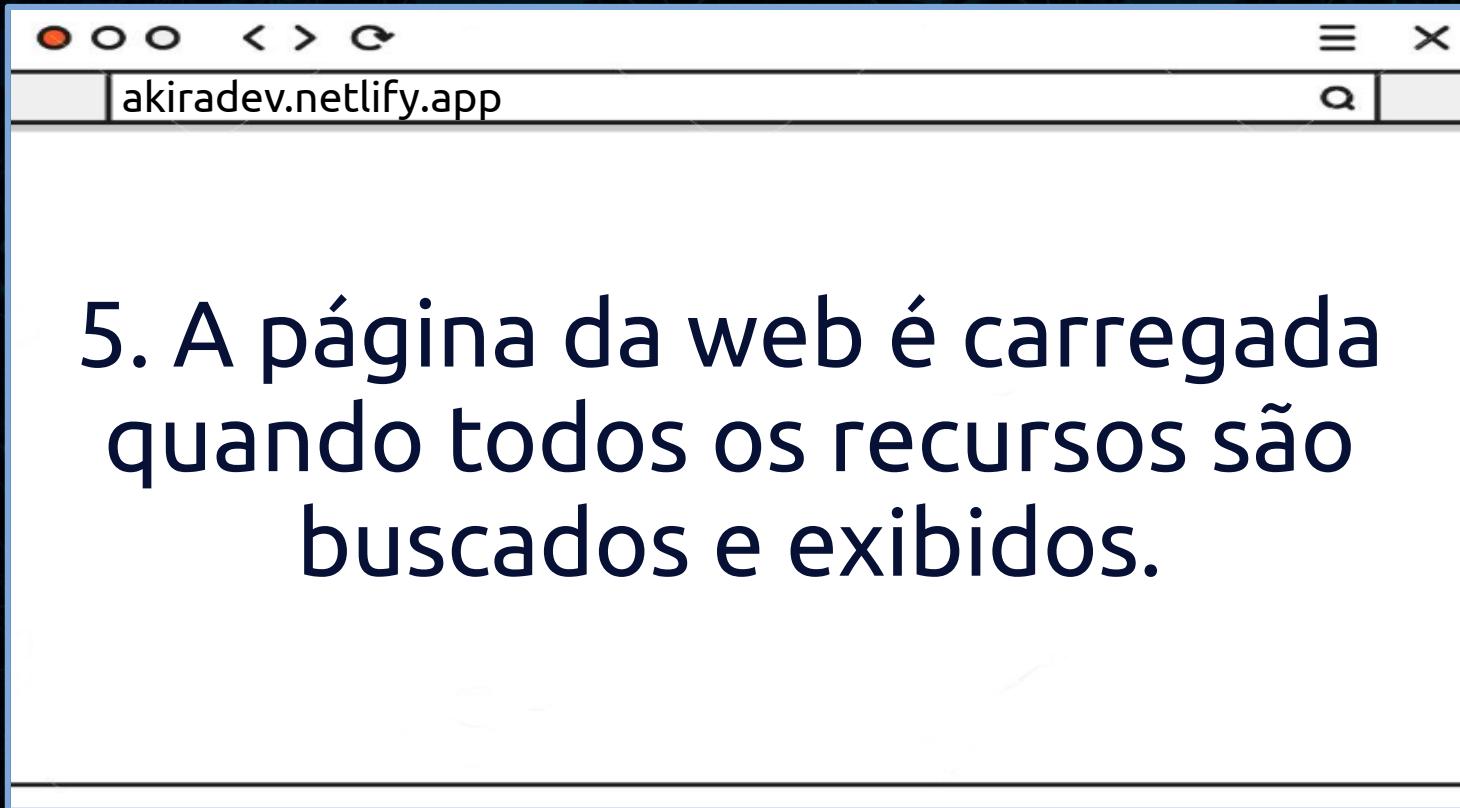
3. O servidor da web responde ao browser com o arquivo HTML que foi solicitado.

Funcionamento



4. O browser lê o HTML, vê os recursos embutidos e pede ao servidor por eles também.

Funcionamento



5. A página da web é carregada quando todos os recursos são buscados e exibidos.

Web Site vs Web Page

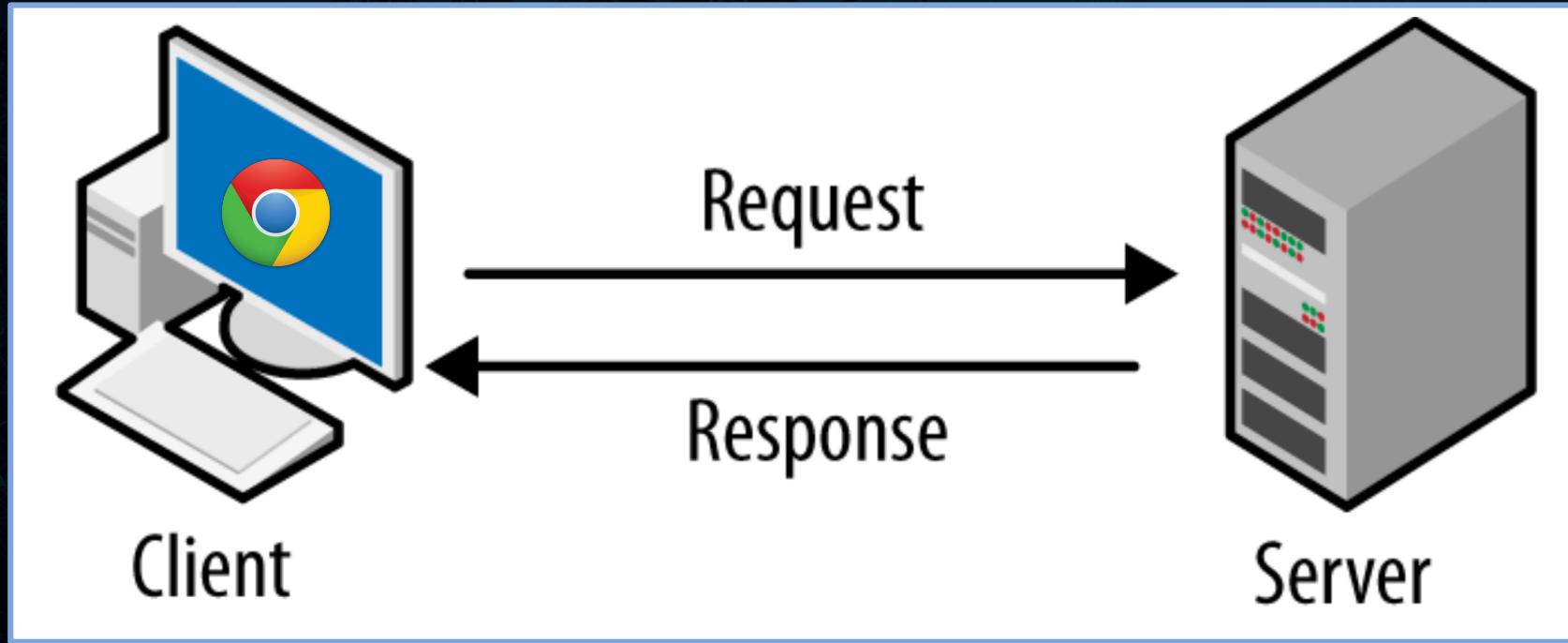
- Um web site refere-se a um local central com mais de uma página web ou uma série de páginas web.
- Por exemplo, <https://www.computerhope.com> é considerado um site que contém milhares de páginas web diferentes.



O Protocolo HTTP

- HTTP significa Hyper Text Transfer Protocol.
- WWW é sobre comunicação entre clientes e servidores web.
- A comunicação entre computadores clientes e servidores web é feita enviando requisições HTTP e recebendo respostas HTTP.
- Os clientes geralmente são navegadores (Chrome, Edge, Safari), mas podem ser qualquer tipo de programa ou dispositivo.
- Os servidores geralmente são computadores na nuvem.

Comunicação Cliente-Servidor



Requisição / Resposta HTTP

- A comunicação entre clientes e servidores é feita por requisições e respostas:
 1. Um cliente (um navegador) envia uma solicitação HTTP para a web.
 2. Um servidor web recebe a requisição.
 3. O servidor executa um aplicativo para processar a solicitação.
 4. O servidor retorna uma resposta HTTP (*saída / output*) para o browser.
 5. O cliente (o navegador) recebe a resposta.

O Círculo de Requisição HTTP

- Um círculo de Requisição / Resposta HTTP:
 1. O navegador solicita uma página HTML. O servidor retorna um arquivo HTML.
 2. O navegador solicita uma folha de estilo. O servidor retorna um arquivo CSS.
 3. O navegador solicita uma imagem JPG. O servidor retorna um arquivo JPG.
 4. O navegador solicita o código JavaScript. O servidor retorna um arquivo JS.
 5. O navegador solicita dados. O servidor retorna dados (em XML ou JSON).

XHR - XML Http Request

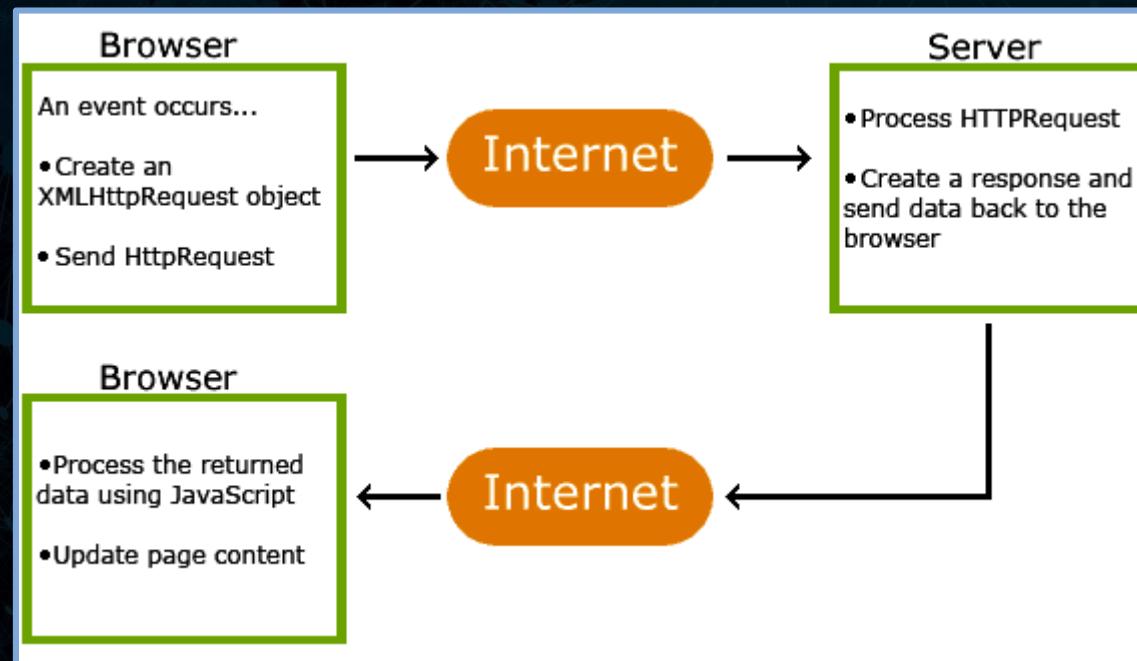
- Todos os navegadores possuem um objeto XMLHttpRequest (XHR) integrado.
- XHR é um objeto JavaScript usado para transferir dados entre um navegador da web e um servidor da web.
- O XHR é freqüentemente usado para solicitar e receber dados com o propósito de modificar uma página da web.

XHR - XML Http Request

- Apesar do XML e Http no nome, o XHR é usado com outros protocolos além do HTTP e os dados podem ser de muitos tipos diferentes, como HTML, CSS, XML, JSON e texto simples.
- O objeto XHR é um sonho de desenvolvedores Web, porque podemos:
 - Atualizar uma página da web sem recarregar a página.
 - Solicitar dados de um servidor - após o carregamento da página.
 - Receber dados de um servidor - após o carregamento da página.
 - Envie dados para um servidor - em segundo plano.

XHR - XML Http Request

- O objeto XHR é o conceito subjacente de AJAX e JSON:

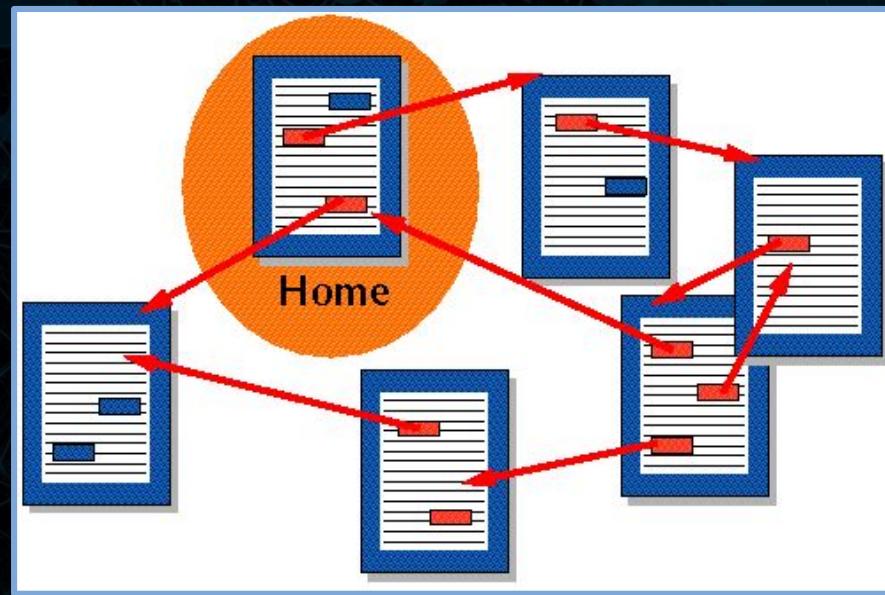


Hipertexto

- Hipertexto é o texto exibido em uma tela de computador ou outros dispositivos eletrônicos com referências (hiperlinks) a outro texto que o leitor pode acessar imediatamente.
- Os documentos de hipertexto são interconectados por hiperlinks, que normalmente são ativados por um clique do mouse, conjunto de teclas ou pelo toque na tela.
- O hipertexto é um dos principais conceitos subjacentes da World Wide Web, onde as páginas da Web são frequentemente escritas em Hypertext Markup Language (HTML).

Hipertexto

- Conforme implementado na Web, o hipertexto permite a publicação fácil de informações na Internet.



Documentos
conectados por
hiperlinks.

HTML

- HTML significa Hyper Text Markup Language.
- HTML é a linguagem de marcação padrão para páginas da web.
- Os elementos HTML são os blocos de construção das páginas HTML.
- Os elementos HTML são representados por <> tags.

Elementos HTML

- Um elemento HTML é uma tag inicial e uma tag final com conteúdo entre:

`<h1>Este é um Título</h1>`

`<p>Este é um Parágrafo</p>`

Tag Inicial	Conteúdo do Elemento	Tag Final
<code><h1></code>	Este é um Título	<code></h1></code>
<code><p></code>	Este é um Parágrafo	<code></p></code>

Atributos HTML

- Elementos HTML podem ter atributos.
- Atributos fornecem informações adicionais sobre o elemento.
- Os atributos vêm em pares nome / valor como charset = "utf-8".
- São valores adicionais que configuram os elementos ou ajustam seu comportamento de várias maneiras para atender aos critérios que os usuários desejam.

Um Simples Documento HTML

```
<!DOCTYPE html>
<html lang="en">

<meta charset="utf-8">
<title>Título da Página</title>

<body>
  <h1>Título do Artigo</h1>
  <p>Parágrafo 1</p>
  <p>Parágrafo 2</p>
</body>

</html>
```

HTML



Exemplo Explicado

- Os elementos HTML são os blocos de construção das páginas HTML.
 - A declaração <!DOCTYPE html> define o documento como sendo HTML5.
 - O elemento <html> é o elemento raiz de uma página HTML.
 - O atributo lang define o idioma do documento.
 - O elemento <meta> contém meta informações sobre o documento.
 - O atributo charset define o conjunto de caracteres usado no documento.
 - O elemento <title> especifica um título para o documento.
 - O elemento <h1> define um título grande.
 - O elemento <p> define um parágrafo.

Documentos HTML

- Todos os documentos HTML devem começar com uma declaração de tipo de documento:
`<!DOCTYPE html>`.
- O próprio documento HTML começa com
`<html>` e termina com `</html>`.
- A parte visível do documento HTML está entre
`<body>` e `</body>`.

Estrutura do Documento HTML

```
<html>
```

```
  <head>
```

```
    <title>Page title</title>
```

```
  </head>
```

```
<body>
```

```
  <h1>This is a heading</h1>
```

```
  <p>This is a paragraph.</p>
```

```
  <p>This is another paragraph.</p>
```

```
</body>
```

```
</html>
```

Cabeçalhos HTML

- Os cabeçalhos HTML são definidos com as tags `<h1>` a `<h6>`.
- `<h1>` define o cabeçalho mais importante. `<h6>` define o título menos importante.

```
<h1>Título 1</h1>
<h2>Título 2</h2>
<h3>Título 3</h3>
```

Parágrafos HTML

- Os parágrafos HTML são definidos com tags `<p>`.

```
<p>Parágrafo 1</p>
<p>Parágrafo 2</p>
<p>Parágrafo 3</p>
<p>Parágrafo 4</p>
```

Links HTML

- Os links HTML são definidos com tags `<a>`.
- O destino do link é especificado no atributo `href`.

```
<a href="https://www.w3schools.com">Aprenda HTML</a>
<a href="https://akiradev.netlify.app">Aprenda Python</a>
<a href="texto.html">Leia o texto</a>
```

Imagens HTML

- As imagens HTML são definidas com tags ``.
- O arquivo de origem (`src`), texto alternativo (`alt`), largura (`width`) e altura (`height`) são fornecidos como atributos.

```


```

Botões HTML

- Os botões HTML são definidos com tags `<button>`.

```
<button>Inserir Dados</button>
<button>Visualizar Dados</button>
<button>Editar Dados</button>
<button>Deletar Dados</button>
```

Listas HTML

- As listas HTML são definidas com tags (lista não ordenada / com marcadores) ou (lista ordenada / numerada), seguidas por tags (itens da lista):

```
<ul>
  <li>Python</li>
  <li>C++</li>
  <li>JavaScript</li>
</ul>
```

Tabelas HTML

- Uma tabela HTML é definida com uma tag `<table>`.
- As linhas da tabela são definidas com tags `<tr>`.
- Os cabeçalhos das tabelas são definidos com tags `<th>`. (negrito e centralizado por padrão).
- As células da tabela (dados) são definidas com tags `<td>`.

Tabela HTML

```
<table>
  <tr>
    <th>Nome</th>
    <th>Sobrenome</th>
  </tr>
  <tr>
    <td>Isaac</td>
    <td>Newton</td>
  </tr>
  <tr>
    <td>Alan</td>
    <td>Turing</td>
  </tr>
</table>
```

Programação HTML

- Todo elemento HTML pode ter atributos.
- Os atributos são sempre especificados na tag inicial.
- Para desenvolvimento e programação web, os atributos mais importantes são id e class.
- Esses atributos são freqüentemente usados para lidar com manipulações programáveis de páginas da web.

Atributos HTML

Atributo	Exemplo
id	<table id="tabela">
class	<p class="normal">
style	<p style="font-size:16px">
data-	<div data-id="300">
onclick	<input onclick="function()">
onmouseover	

Todas as Tags HTML



<https://allthetags.com>

Tutorial Completo HTML



<https://www.w3schools.com/html/default.asp>

<https://developer.mozilla.org/en-US/docs/Learn/HTML>

CSS

- CSS significa Cascading Style Sheets.
- CSS é o código que estiliza o conteúdo da web.
- CSS descreve como os elementos HTML devem ser exibidos.
- Como o HTML, o CSS não é uma linguagem de programação. Também não é uma linguagem de marcação. CSS é uma linguagem de folha de estilo.

CSS

- CSS é o que usamos para estilizar elementos HTML seletivamente. Por exemplo, este CSS seleciona o texto de um parágrafo, definindo a sua cor para vermelho:

```
p {  
    color: red;  
}
```



Linkando CSS com HTML

- Uma folha de estilo CSS pode ser armazenada em um arquivo externo.
- Usando um editor de texto, cole as três linhas de CSS (anteriores) em um novo arquivo. Salve o arquivo como style.css em um diretório denominado styles.
- Folhas de estilo externas são vinculadas a páginas HTML com tags <link>.

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

Sintaxe CSS

- Uma regra CSS consiste em um seletor e um bloco de declaração.



```
p {  
    color: red;  
}
```

The diagram illustrates the structure of a CSS rule. It starts with a **Selector** (`p`) pointing to the element name. This is followed by a **Brace** (`{`) which encloses the **Declaration**. Inside the brace, there is a **Property** (`color`) and its corresponding **Value** (`red`). The entire block of properties and values is labeled as the **Declaration**.

Sintaxe CSS

- Toda a estrutura é chamada de conjunto de regras. Observe os nomes das partes individuais:
 - Seletor: Este é o nome do elemento HTML no início do conjunto de regras. Define o(s) elemento(s) a serem estilizados. Para estilizar um elemento diferente, altere o seletor.
 - Declaração: Esta é uma regra única, como `color: red;`. Ele especifica quais propriedades do elemento você deseja estilizar.
 - Propriedades: Essas são maneiras de definir o estilo de um elemento HTML. Em CSS, você escolhe quais propriedades deseja afetar na regra.
 - Valor da Propriedade: À direita da propriedade - após os dois pontos - está o valor da propriedade. Isso escolhe uma das muitas aparências possíveis para uma determinada propriedade. (Por exemplo, existem muitos valores de cores além do vermelho.)

Sintaxe CSS

- Observe as outras partes relevantes da sintaxe:
 - Além do seletor, cada conjunto de regras deve ser colocado entre chaves. ({})
 - Em cada declaração, você deve usar dois-pontos (:) para separar a propriedade de seu valor ou valores.
 - Em cada conjunto de regras, você deve usar um ponto-e-vírgula (;) para separar cada declaração da próxima.

Sintaxe CSS

- Para modificar vários valores de propriedade em um conjunto de regras, devemos escrever eles separados por ponto e vírgula, assim:

```
p {  
    color: red;  
    width: 500px;  
    border: 1px solid black;  
}
```

Selecionando Múltiplos Elementos

- Podemos também selecionar vários elementos e aplicar um único conjunto de regras a todos eles.
- Separamos vários seletores por vírgulas. Por exemplo:

```
p, li, h1 {  
    color: red;  
}
```

Diferentes Tipos de Seletores

- Existem muitos tipos diferentes de seletores.
- Os exemplos anteriores usam seletores de elemento, que selecionam todos os elementos de um determinado tipo.
- Mas podemos fazer seleções mais específicas também.
- A seguir estão alguns dos tipos mais comuns de seletores.

Seletor de Elemento

- Seletor de elemento (às vezes chamado de seletor de tag ou seletor de tipo).
- É responsável por selecionar todos os elementos HTML do tipo especificado.
- Por exemplo, p seleciona os elementos <p>.

```
p {  
    color: red;  
}
```

Seletor de ID

- O seletor de ID é capaz de selecionar o elemento na página com o ID especificado. Em uma determinada página HTML, cada valor de id deve ser único.
- `#my-id` seleciona `<p id="my-id">` ou ``.

```
#my-id {  
    font-size: 20px;  
}
```

Seletor de Classe

- O seletor de Classe é capaz de selecionar o(s) elemento(s) na página com a classe especificada. Várias instâncias da mesma *class* podem aparecer em uma página.
- .my-class seleciona <p class="my-class"> e .

```
.my-class {  
    font-weight: bold;  
}
```

Seletor de Atributo

- O seletor de Atributo é capaz de selecionar o(s) elemento(s) na página com o atributo especificado.
- `img[src]` seleciona `` mas não ``.

```
img[src] {  
    width: 250px;  
}
```

Seletor Pseudo-class

- O seletor Pseudo-class é capaz de selecionar o(s) elemento(s) especificado(s), mas apenas quando no estado especificado. (Por exemplo, quando um cursor passa sob um link.).
- a:hover seleciona <a>, mas apenas quando o ponteiro do mouse estiver pairando sobre o link.

```
a:hover {  
    color: green;  
}
```

CSS: Tudo sobre Caixas

- Algo que você notará sobre como escrever CSS: muito disso é sobre caixas.
- Isso inclui a configuração de tamanho, cor e posição.
- A maioria dos elementos HTML em sua página pode ser considerada como caixas colocadas em cima de outras caixas.
- O layout CSS é baseado principalmente no *box model*.

Box Model

- Cada caixa que ocupa espaço em sua página tem propriedades como:
 - padding: o espaço ao redor do conteúdo. No exemplo a seguir, é o espaço ao redor do texto do parágrafo.
 - border: a linha sólida que está fora do padding.
 - margin: o espaço ao redor da borda.

Box Model

margin

border

padding

 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla id neque. Etiam vestibulum, augue sit amet condimentum imperdiet, diam neque blandit lacus, venenatis ultrices nunc lorem laoreet orci. Integer tortor urna, viverra in, egestas at, volutpat vel, nunc. Donec eget ipsum. Cras lacus. Nunc egestas ligula quis purus. Etiam viverra dignissim erat. Pellentesque facilisis. Quisque et eros eget ante condimentum.

CSS Interno

- Uma folha de estilo interna pode ser usada se uma única página HTML tiver um estilo exclusivo.
- O estilo interno é definido dentro do elemento `<style>`, dentro da seção head.

CSS Interno

```
<!DOCTYPE html>
<html>
<style>
body {
    background-color: orange;
}
</style>
<body>

<h1>CSS Interno</h1>

</body>
</html>
```

CSS Inline

- Um estilo inline pode ser usado para aplicar um estilo único a um único elemento.
- Para usar estilos inline, adicione o atributo de estilo ao elemento relevante. O atributo style pode conter qualquer propriedade CSS.

CSS Inline

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">CSS</h1>
<p style="color:red;">Inline</p>

</body>
</html>
```

Layout CSS - A propriedade display

- A propriedade display é a propriedade CSS mais importante para controlar o layout.
- A propriedade display especifica se e como um elemento é exibido.
- Cada elemento HTML tem um valor de exibição (*display*) padrão dependendo de que tipo de elemento ele é. O valor de exibição padrão para a maioria dos elementos é block ou inline.

Elementos Block-level

- Um elemento block-level sempre começa em uma nova linha e ocupa toda a largura disponível (estende-se para a esquerda e para a direita, tanto quanto pode).
- O elemento `<div>` é um elemento block-level.
- Exemplos de elementos block-level: `<h1>` - `<h6>`, `<p>`, `<form>`, `<header>`, `<footer>`, `<section>`.

Elementos Inline

- Um elemento inline não começa em uma nova linha e só ocupa a largura necessária.
- Exemplos de elementos inline:
 - ``
 - `<a>`
 - ``
 - `<button>`
 - ``

Display: none;

- `display: none;` é comumente usado com JavaScript para ocultar e mostrar elementos sem excluí-los e recriá-los.
- O elemento `<script>` usa `display: none;` como padrão.

Substituir o Valor de Exibição Padrão

- Conforme mencionado, cada elemento tem um valor de exibição padrão. No entanto, podemos substituir isso.
- Alterar um elemento inline para um elemento block, ou vice-versa, pode ser útil para fazer a página parecer de uma maneira específica e ainda seguir os padrões da web.
- Um exemplo comum é criar elementos inline para menus horizontais:

```
li {  
    display: inline;  
}
```

Visibilidade

- Ocultar um elemento pode ser feito definindo a propriedade `display` como `none`. O elemento ficará oculto e a página será exibida como se o elemento não estivesse lá.
- `visibility:hidden;` também oculta um elemento. No entanto, o elemento ainda ocupará o mesmo espaço de antes. O elemento ficará escondido, mas ainda afetará o layout.

```
h1 {  
    visibility: hidden;  
}
```

Ordem em Cascata

- Se estilos diferentes forem especificados para elementos HTML, os estilos irão se cascatear em novos estilos com a seguinte prioridade:
 - Prioridade 1: estilos inline.
 - Prioridade 2: folhas de estilo externas e internas.
 - Prioridade 3: padrão do browser.
 - Se estilos diferentes forem definidos no mesmo nível de prioridade, o último terá a prioridade mais alta.

Tutorial Completo CSS



<https://www.w3schools.com/css/default.asp>

https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps

Seletor de Cores



<https://htmlcolorcodes.com/color-picker>

Web Design Responsivo

- O Responsive Web Design trata do uso de HTML e CSS para redimensionar automaticamente um site.
- O Responsive Web Design trata de fazer um site ter uma boa aparência em todos os dispositivos (desktops, tablets e telefones).

CONTENT IS LIKE WATER



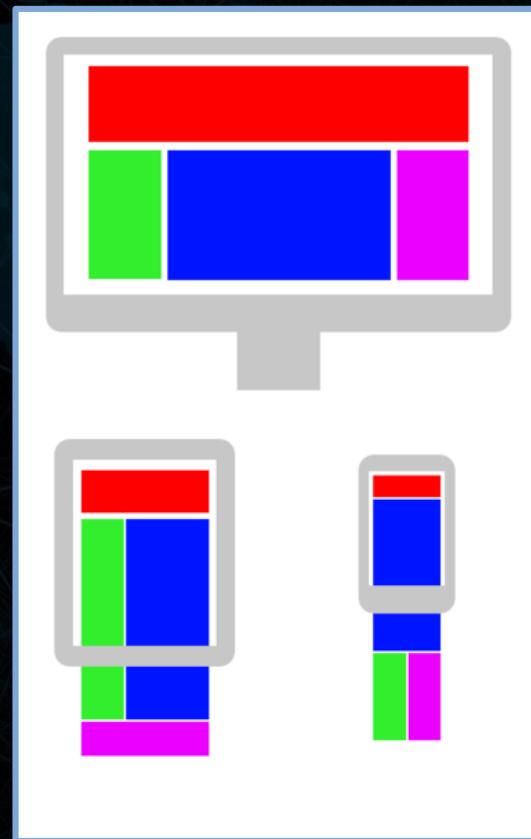
“ You put water into a cup it becomes the cup.
You put water into a bottle it becomes the bottle.
You put it in a teapot, it becomes the teapot. ”

Josh Clark (originally Bruce Lee) - Seven deadly mobile myths

Illustration by Stéphanie Walter

Web Design Responsivo

- Um exemplo de como vários elementos de uma página da web se adaptam ao tamanho da tela de diferentes dispositivos, como a tela de um computador desktop, tablet PC e um smartphone.
- O design responsivo da web se tornou mais importante, pois a quantidade de tráfego móvel passou a representar mais da metade do tráfego total da Internet.



Configurando o Viewport

- Ao criar páginas da web responsivas, devemos adicionar o seguinte elemento <meta> a todas as nossas páginas da web.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Media Queries

- As Media Queries desempenham um papel importante em páginas web responsivas.
- As Media Queries são um recurso do CSS3 que permite a renderização de conteúdo para se adaptar a diferentes condições, como a resolução da tela (por exemplo, tamanho da tela do celular e desktop).
- Com as Media Queries, podemos definir estilos diferentes para diferentes tamanhos de navegador.

Imagens Responsivas

- Imagens responsivas são imagens que se adaptam perfeitamente a qualquer tamanho de navegador.
- Quando a propriedade width CSS é definida como um valor de porcentagem, uma imagem aumentará ou diminuirá ao redimensionar a janela do navegador.

```

```

Imagens Responsivas

- Se a propriedade max-width for definida como 100%, a imagem será reduzida se necessário, mas nunca aumentará para ser maior do que seu tamanho original.

```

```

Imagen Dependendo do Tamanho do Browser

- O elemento HTML <picture> permite definir diferentes imagens para diferentes tamanhos de janela do navegador.

```
<picture>
  <source srcset="img1.jpg" media="(max-width: 600px)">
  <source srcset="img2.jpg" media="(max-width: 1500px)">
  <source srcset="img3.jpg">
  
</picture>
```

Responsive W3.CSS

- W3.CSS é um framework CSS gratuito que oferece design responsivo por padrão.
- O W3.CSS facilita o desenvolvimento de sites que tenham uma boa aparência em qualquer dispositivo: desktop, laptop, tablet ou telefone.
- Para aprender mais sobre W3.CSS visite:
<https://www.w3schools.com/w3css/default.asp>

Bootstrap

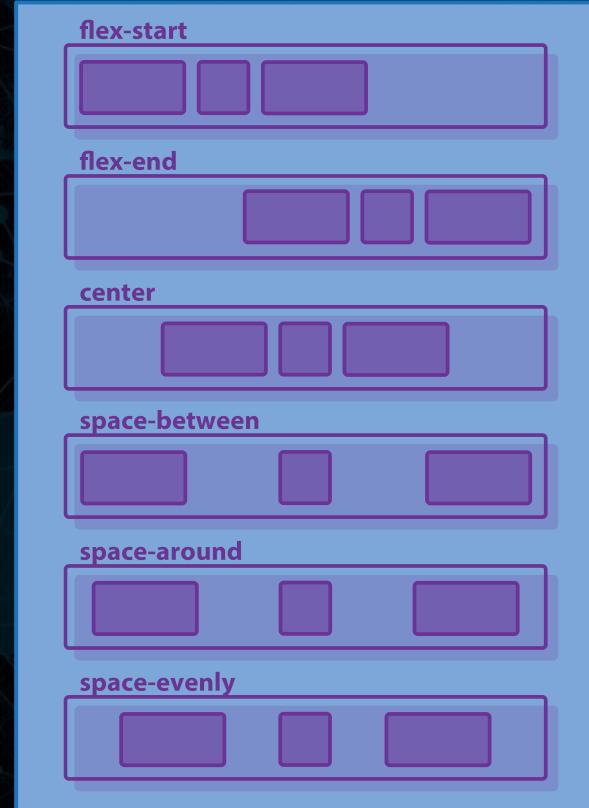
- Bootstrap é um framework muito popular que usa HTML, CSS e JavaScript para criar páginas web responsivas.
- Para aprender mais sobre W3.CSS visite:
- <https://www.w3schools.com/bootstrap4/default.asp>

B

Flexbox Layout

- O módulo Flexbox Layout (Flexible Box) visa fornecer uma maneira mais eficiente de dispor, alinhar e distribuir espaço entre os itens em um contêiner, mesmo quando seu tamanho é desconhecido e / ou dinâmico (daí a palavra “flex”).
- A ideia principal por trás do layout flexível é dar ao contêiner a capacidade de alterar a largura / altura (e ordem) de seus itens para melhor preencher o espaço disponível (principalmente para acomodar todos os tipos de dispositivos de exibição e tamanhos de tela). Um contêiner flexível expande os itens para preencher o espaço livre disponível ou os reduz para evitar o transbordamento.

Flexbox Layout



Grid Layout

- O CSS Grid Layout Module oferece um sistema de layout baseado em grade, com linhas e colunas, facilitando o design de páginas web sem ter que usar floats e posicionamento.



JavaScript

- JavaScript é a linguagem de programação para a web.
- JavaScript pode atualizar e alterar HTML e CSS.
- JavaScript pode calcular, manipular e validar dados.

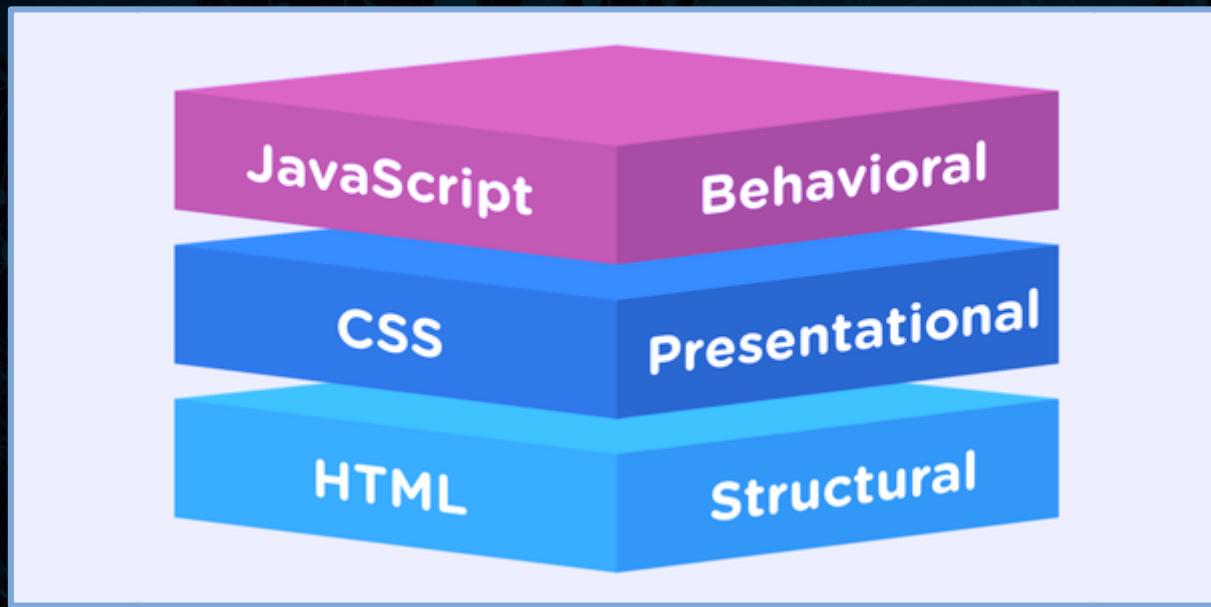


JavaScript

- JavaScript é a terceira camada das tecnologias da Web padrão, incluindo (HTML e CSS).
 - HTML é a linguagem de marcação que usamos para estruturar e dar significado ao nosso conteúdo da web, por exemplo, definir parágrafos, cabeçalhos e tabelas de dados, ou incorporar imagens e vídeos na página.
 - CSS é uma linguagem de regras de estilo que usamos para aplicar estilos ao nosso conteúdo HTML, por exemplo, definindo cores e fontes de fundo e organizando nosso conteúdo em várias colunas.
 - JavaScript é uma linguagem de script que permite criar conteúdo de atualização dinâmica, controlar multimídia, animar imagens e praticamente tudo que quisermos.

JavaScript

- As três camadas são construídas uma sobre a outra muito bem.



O Poder de JavaScript

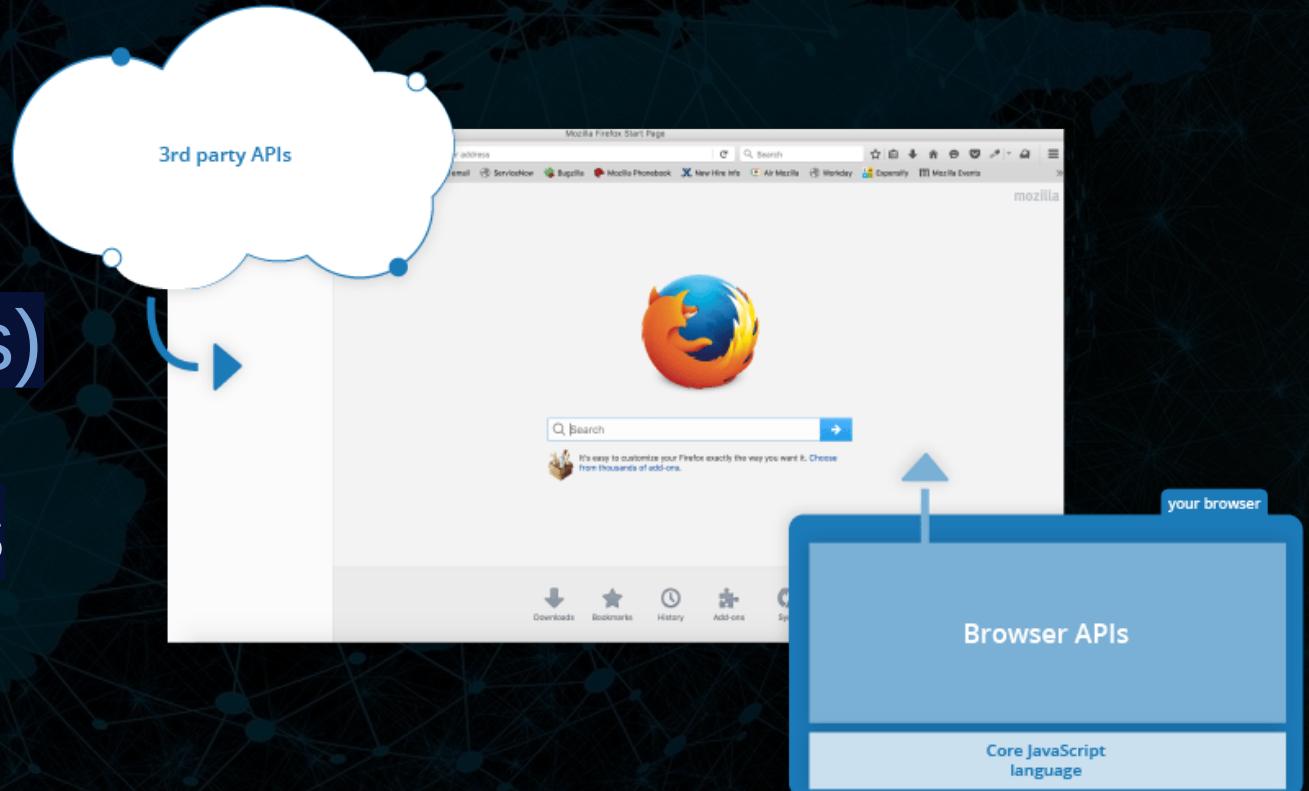
- O núcleo principal client-side da linguagem JavaScript consiste em alguns recursos de programação comuns que permitem fazer coisas como:
 - Alterar o conteúdo HTML.
 - Alterar valores de atributo HTML.
 - Alterar estilos HTML (CSS).
 - Ocultar elementos HTML.
 - Mostrar elementos HTML.

Application Programming Interfaces

- O que é ainda mais emocionante, entretanto, é a funcionalidade construída sob o client-side da linguagem JavaScript.
- As chamadas Application Programming Interfaces (APIs) fornecem superpoderes extras para usar em nosso código JavaScript.
- APIs são conjuntos prontos de blocos de construção de código que permitem a um desenvolvedor implementar programas que, de outra forma, seriam difíceis ou impossíveis de implementar.

Application Programming Interfaces

- Application Programming Interfaces (APIs) geralmente estão em duas categorias.



Browser APIs

- As APIs do navegador são integradas ao seu navegador da web e são capazes de expor dados do ambiente de computador ou fazer coisas complexas úteis. Por exemplo:
 - A DOM (Document Object Model) API permite que você manipule HTML e CSS, criando, removendo e alterando HTML, aplicando dinamicamente novos estilos à sua página, etc.
 - A Geolocation API recupera informações geográficas.
 - As APIs Canvas e WebGL permitem criar gráficos 2D e 3D animados.
 - APIs de áudio e vídeo como HTMLMediaElement e WebRTC permitem que você faça coisas realmente interessantes com multimídia, como reproduzir áudio e vídeo diretamente em uma página da web.

Third Party APIs

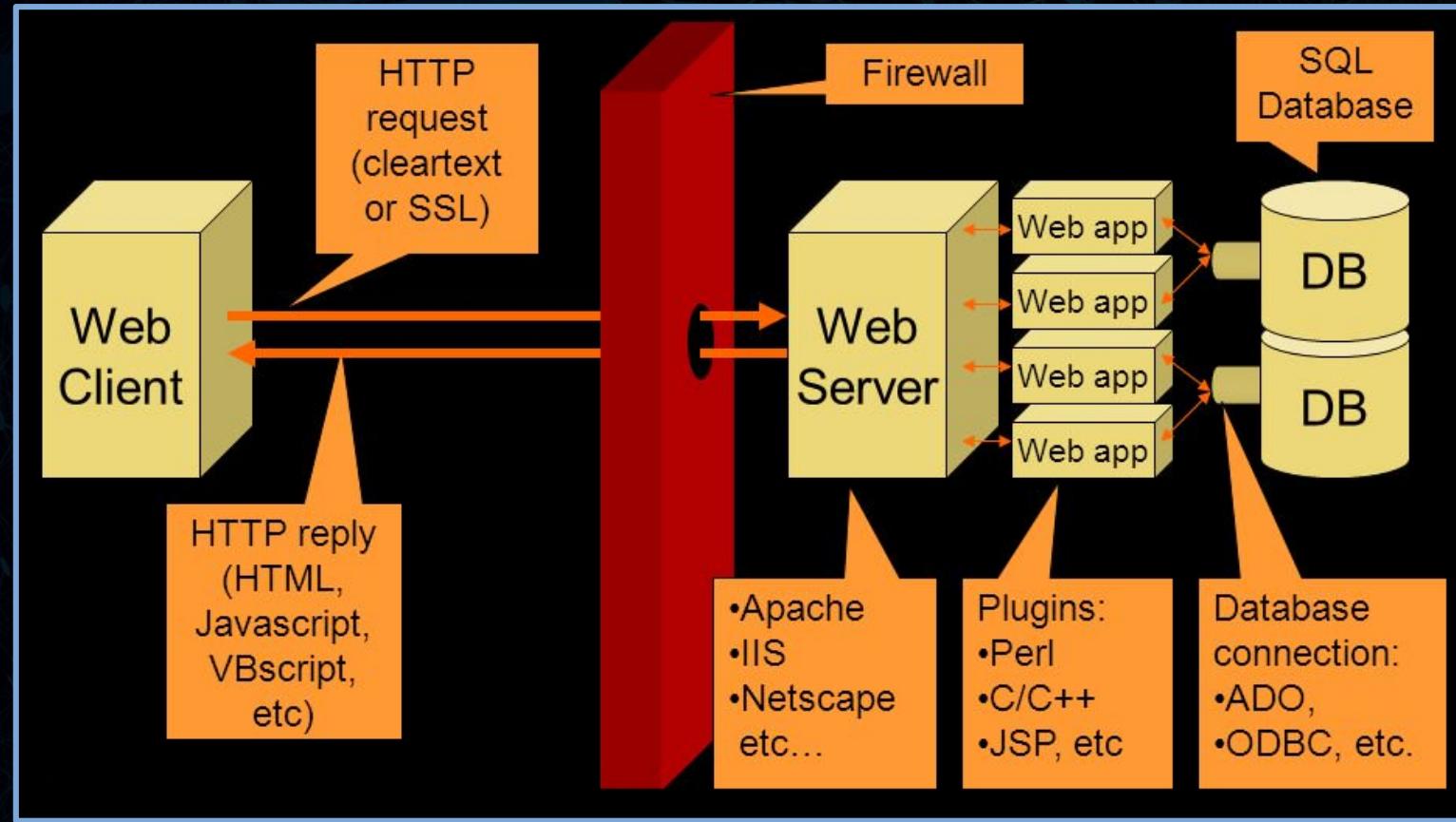
- APIs de terceiros não são incorporadas ao navegador por padrão, e geralmente você precisa obter seu código e informações de algum lugar da web. Por exemplo:

- A API do Twitter permite que você faça coisas como exibir seus tweets mais recentes em seu site.
- A API do Google Maps e a API do OpenStreetMap permitem que você incorpore mapas personalizados em seu site e outras funcionalidades semelhantes.

Client-side vs Server-side

- Você já deve ter ouvido os termos código do lado do servidor e código do lado do cliente, especialmente no contexto de desenvolvimento da web.
- O código do lado do cliente é o código que é executado no computador do usuário - quando uma página da web é exibida, o código do lado do cliente da página é baixado, executado e exibido pelo navegador.
- O código do lado do servidor, por outro lado, é executado no servidor e, em seguida, seus resultados são baixados e exibidos no navegador. Exemplos de linguagens da web populares do lado do servidor incluem: PHP, Python, Ruby, ASP.NET e JavaScript!

Client-side vs Server-side



Código Dinâmico

- A palavra dinâmica é usada para descrever JavaScript do lado do cliente e linguagens do lado do servidor.
- Refere-se à capacidade de atualizar a exibição de uma página da web para mostrar coisas diferentes em circunstâncias diferentes, gerando novos conteúdos conforme necessário.
- O código do lado do servidor gera dinamicamente novo conteúdo no servidor, por exemplo, puxando dados de um banco de dados.
- Enquanto o JavaScript do lado do cliente gera dinamicamente novo conteúdo dentro do navegador no cliente, por exemplo, criando uma nova tabela HTML, preenchendo-a com os dados solicitados do servidor e, em seguida, exibindo a tabela.

Configurando o JavaScript

- O JavaScript é aplicado à sua página HTML de maneira semelhante ao CSS.
- Enquanto CSS usa elementos `<link>` para aplicar folhas de estilo externas e elementos `<style>` para aplicar folhas de estilo internas ao HTML.
- JavaScript só precisa de uma tag no mundo do HTML - o elemento `<script>`.

JavaScript Interno

- Em HTML, o código JavaScript é inserido entre as tags <script> e </script>.
- Você pode colocar qualquer número de scripts em um documento HTML.
- Os scripts podem ser colocados em <body>, ou na seção <head> de uma página HTML, ou em ambos.

```
<script>
  console.log("Hello World");
</script>
```

JavaScript Externo

- E se quisermos colocar nosso JavaScript em um arquivo externo?
- Primeiro, crie um novo arquivo no mesmo diretório do arquivo HTML de amostra. Chame-o de script.js - certifique-se de que tem a extensão de nome de arquivo .js, pois é assim que é reconhecido como JavaScript.
- Substitua seu elemento <script> atual pelo seguinte:

```
<script src="script.js"></script>
```

Carregando Scripts

- Existem vários problemas envolvidos em fazer com que os scripts sejam carregados no momento certo.
- Nada é tão simples quanto parece! Um problema comum é que todo o HTML de uma página é carregado na ordem em que aparece.
- Se você estiver usando JavaScript para manipular elementos na página (ou mais precisamente, o Document Object Model), seu código não funcionará se o JavaScript for carregado e analisado antes do HTML que você está tentando manipular.

Carregando Scripts

- Uma maneira de corrigir esses possíveis problemas é usando um event listener, que escuta o evento "DOMContentLoaded" do navegador, o que significa que o corpo HTML está completamente carregado e analisado.
- O JavaScript dentro deste bloco não será executado até que o evento seja disparado, portanto, o erro é evitado.

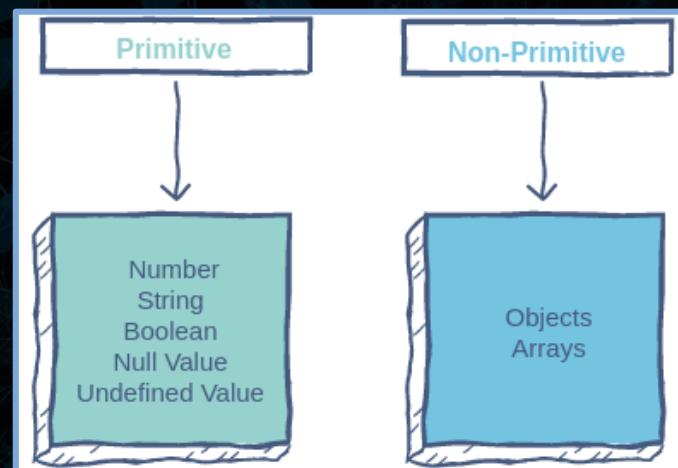
```
document.addEventListener("DOMContentLoaded", function() {  
    ...  
});
```

Visão Geral

- JavaScript é uma linguagem multiparadigma, dinâmica, com tipos e operadores, objetos embutidos padrão e métodos.
- Sua sintaxe é baseada nas linguagens Java e C - muitas estruturas dessas linguagens também se aplicam ao JavaScript.
- JavaScript suporta programação orientada a objetos com protótipos de objetos.
- O JavaScript também suporta programação funcional - por serem objetos, as funções podem ser armazenadas em variáveis e passadas como qualquer outro objeto.

Tipos

- Começamos examinando os blocos de construção de qualquer linguagem: os tipos.
- Os programas JavaScript manipulam valores, e todos esses valores pertencem a um tipo. Os tipos de JavaScript são:
 - Number
 - Object
 - null
 - undefined
 - String
 - Boolean
 - Symbol



Números

- Os números em JavaScript são "valores IEEE 754 no formato de 64 bits de dupla precisão", de acordo com a especificação - Não existem números inteiros em JavaScript (exceto BigInt).

```
console.log(3 / 2); // 1.5  
console.log(Math.floor(3 / 2)); // 1
```

- Portanto, um número inteiro aparente é de fato implicitamente um float.

Números

- Os operadores aritméticos padrão são suportados, incluindo adição, subtração, módulo (ou resto) e assim por diante.
- Há também um objeto integrado chamado Math, que fornece funções matemáticas avançadas e constantes:

```
var r = 5;  
Math.sin(3.5);  
let circumference = 2 * Math.PI * r;
```

Números

- Podemos converter uma string em um inteiro usando a função interna parseInt().
- Ela toma a base para a conversão como um segundo argumento opcional, que você sempre deve fornecer:

```
parseInt('333', 10); // 333  
parseInt('111', 2); // 7  
parseInt('ff', 16); // 255
```

Strings

- Strings em JavaScript são sequências de caracteres Unicode.
- Mais precisamente, são sequências de unidades de código UTF-16, cada unidade de código é representada por um número de 16 bits.
- Cada caractere Unicode é representado por 1 ou 2 unidades de código.
- Para encontrar o comprimento de uma string (em unidades de código), acessamos sua propriedade de `length`.

```
"gabriel".length; // 7
```

Strings

- Esse é o nosso primeiro experimento com objetos JavaScript!
- Mencionamos que você também pode usar strings como objetos?
- Eles também têm métodos que permitem manipular a string e acessar informações sobre a string.

```
'gabriel'.charAt(0) // 'g'  
"hello world".replace("world","gabriel") // 'hello gabriel'  
'javascript'.toUpperCase(); // 'JAVASCRIPT'
```

Outros Tipos

- JavaScript distingue entre null, que é um valor que indica um não valor deliberado (e só é acessível por meio da palavra-chave null), e undefined, que é um valor do tipo indefinido que indica uma variável não inicializada - isto é, um valor não atribuído ainda.
- em JavaScript, é possível declarar uma variável sem atribuir um valor a ela. Se você fizer isso, o tipo da variável será undefined.
- undefined é na verdade uma constante.

Outros Tipos

- JavaScript tem um tipo boolean, com valores possíveis true e false (ambos são palavras-chave). Qualquer valor pode ser convertido em booleano de acordo com as seguintes regras:
 1. false, 0, strings vazias (""), NaN, null e undefined tornam-se todos false.
 2. Todos os outros valores se tornam true.

```
Boolean(""); // false  
Boolean(234); // true  
Boolean(undefined); // false
```

Variáveis

- Novas variáveis em JavaScript são declaradas usando uma das três palavras-chave: let, const ou var.
- let nos permite declarar variáveis em nível de bloco. A variável declarada está disponível no bloco em que está incluída.

```
let a;  
let nome = 'Python';
```

Variáveis

- `const` nos permite declarar variáveis cujos valores nunca devem ser alterados. A variável está disponível no bloco em que é declarada.

```
const Pi = 3.14; // variável Pi é declarada  
Pi = 1; // irá gerar um erro
```

- `var` é a palavra-chave declarativa mais comum. Ele não tem as restrições que as outras duas palavras-chave têm.
- Uma variável declarada com a palavra-chave `var` está disponível na função em que é declarada.

```
var x;  
x = 15;
```

Variáveis

- Uma diferença importante entre JavaScript e outras linguagens como Java é que em JavaScript os blocos não têm escopo; apenas funções têm um escopo.
- Portanto, se uma variável for definida usando var em uma instrução composta (por exemplo, dentro de uma estrutura de controle if), ela será visível para toda a função. No entanto, a partir do ECMAScript 2015, as declarações let e const permitem que você crie variáveis com escopo de bloco.

Operadores

- Os operadores numéricos do JavaScript são +, -, *, / e %, que é o operador de resto (módulo).
- Os valores são atribuídos usando = e também há instruções de atribuição compostas, como += e -=.

```
let y = 6;  
y = y + 4;  
y -= 4;
```

Operadores

- Podemos usar ++ e -- para aumentar e diminuir respectivamente. Eles podem ser usados como operadores de prefixo ou postfix.

```
--y;
```

```
y++;
```

- O operador + também faz concatenação de string.

```
"hello" + " world"; // 'hello world'
```

Comparações

- As comparações em JavaScript podem ser feitas usando `<`, `>`, `<=`, `>=`, `!=` e `!==`.
- Elas funcionam para strings e números.
- A igualdade é um pouco menos intuitiva. O operador double-equals executa coerção de tipo se você atribuir tipos diferentes, com resultados às vezes interessantes:

```
123 == '123'; // true  
1 == true; // true
```

Para evitar a coerção de tipo, use o operador triplo-igual. (`====`)

Estruturas de Controle

- JavaScript tem um conjunto semelhante de estruturas de controle a outras linguagens da família C. As instruções condicionais são suportadas por if e else.

```
let num = 20;

if (num == 10) {
    console.log('Número igual a 10')
} else if (num < 10) {
    console.log('Número menor que 10')
} else {
    console.log('Número maior que 10') // Número maior que 10
}
```

Estruturas de Controle

- JavaScript tem loops while e loops do-while. O primeiro é bom para loop básico; o segundo para loops onde você deseja garantir que o corpo do loop seja executado pelo menos uma vez.

```
while (true) {  
    // um loop infinito!  
}  
  
var input;  
do {  
    input = get_input();  
} while (inputIsValid(input));
```

Estruturas de Controle

- JavaScript for loop é o mesmo que em C e Java: ele permite que você forneça as informações de controle para seu loop em uma única linha.

```
for (var i = 0; i < 5; i++) {  
    // Executará 5 vezes  
}
```

- JavaScript também contém dois outros for loops proeminentes.

Estruturas de Controle

- **for ... of:**

```
for (let valor of array) {  
    // Faça algo com valor  
}
```

- **for ... in:**

```
for (let propriedade in object) {  
    // Faça algo com a propriedade do objeto  
}
```

Estruturas de Controle

- Os operadores `&&` e `||` usam lógica de circuito, o que significa que a execução do segundo operando depende do primeiro.

```
0 > 1 && 1 > 0 // false  
0 > 1 || 1 > 0 // true
```

- JavaScript tem um operador ternário para expressões condicionais.

```
var permitido = (idade > 18) ? 'sim' : 'não';
```

Estruturas de Controle

- A instrução switch pode ser usada para vários casos com base em um número ou string.

```
switch(mês) {  
    case 'janeiro':  
        console.log('Bom começo de ano')  
        break  
    case 'dezembro':  
        console.log('Bom fim de ano')  
        break  
    default:  
        console.log('Mês inválido')  
}
```

Objetos

- Os objetos JavaScript podem ser considerados coleções simples de pares nome-valor. Como tal, são semelhantes a:
 - Dicionários em Python.
 - Hashes em Perl e Ruby.
 - Hash Tables em C e C++.
 - HashMaps em Java.
 - Arrays Associativos em PHP.

Objetos

- O fato dessa estrutura de dados ser amplamente usada é uma prova de sua versatilidade.
- Como tudo em JavaScript é um objeto, qualquer programa JavaScript naturalmente envolve muitas pesquisas em tabelas hash.
- A parte "nome" é uma string JavaScript, enquanto o valor pode ser qualquer valor JavaScript - incluindo mais objetos. Isso permite que você crie estruturas de dados de complexidade arbitrária.

Objetos

- Existem duas maneiras básicas de criar um objeto vazio.

```
var obj = new Object();
```

- A outra maneira é:

```
var obj = {};
```

- Eles são semanticamente equivalentes; a segunda é chamada de sintaxe literal de objeto e é mais conveniente.

Objetos

- A sintaxe literal do objeto pode ser usada para inicializar um objeto em sua totalidade.

```
var obj = {  
    nome: 'Tomate',  
    detalhes: {  
        cor: 'vermelho',  
        tamanho: 10  
    }  
};
```

Objetos

- O acesso aos atributos pode ser encadeado:

```
console.log(obj.nome); // Tomate  
console.log(obj['nome']) // Tomate  
console.log(obj.detalhes.cor); // vermelho  
console.log(obj['detalhes']['tamanho']); // 10
```

Objetos

- O exemplo a seguir cria um protótipo de objeto (Pessoa) e uma instância desse protótipo (pessoa).

```
function Pessoa(nome, idade) {  
    this.nome = nome;  
    this.idade = idade;  
}
```

```
// Define um objeto  
var pessoa = new Pessoa('Yuri', 5);  
// Nós estamos criando uma nova pessoa de nome "Yuri" com idade 5.
```

Arrays

- Arrays em JavaScript são, na verdade, um tipo especial de objeto.
- Eles funcionam de forma muito semelhante a objetos regulares (propriedades numéricas podem naturalmente ser acessadas apenas usando a sintaxe []), mas eles têm uma propriedade mágica chamada 'length'.
- O 'length' é sempre um a mais do que o índice mais alto do array.

```
var a = new Array();
a[0] = 'cachorro';
a[1] = 'gato';
a[2] = 'lagarto'; // índice mais alto
a.length; // 3
```

Arrays

- Uma notação mais conveniente para definir um array é usar um literal de array:

```
var a = ['cachorro', 'gato', 'lagarto'];
```

- Podemos iterar sob um array usando o for loop:

```
for (var i = 0; i < a.length; i++) {  
    // Faça algo com a[i]  
}
```

Arrays

- ES2015 introduziu o `for ... of` loop, mais conciso para objetos iteráveis, como arrays:

```
for (const valor of a) {  
    // Faça algo com valor  
}
```

- Outra maneira de iterar sob um array é o `forEach()`:

```
[11, 22, 33].forEach(function(valor, indice, array) {  
    // Fala algo com valor ou array[indice]  
});
```

Arrays

- Se você deseja anexar um item a um array, faça assim:

```
a.push(item);
```

- Os arrays vêm com vários métodos, você pode ver todos eles na documentação completa:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array



Funções

- Junto com os objetos, as funções são o componente principal na compreensão do JavaScript.

```
function adicionar(x, y) {  
    var total = x + y;  
    return total;  
}
```

- Isso demonstra uma função básica. Uma função JavaScript pode ter 0 ou mais parâmetros nomeados.

Funções

- O corpo da função pode conter quantas instruções você quiser e pode declarar suas próprias variáveis que são locais para essa função.
- A instrução return pode ser usada para retornar um valor a qualquer momento, encerrando a função.
- Se nenhuma instrução de retorno for usada (ou um retorno vazio, sem valor), o JavaScript retornará undefined.

Funções

- Os parâmetros nomeados revelam-se mais como diretrizes de uso da função.
- Você pode chamar uma função sem passar os parâmetros esperados, nesse caso, eles serão definidos como `undefined`.
- Você também pode passar mais argumentos do que a função espera.

```
console.log(adicionar()) // NaN  
console.log(adicionar(2,3)) // 5  
console.log(adicionar(2,3,4)) // 5
```

Funções

- As funções têm acesso a uma variável adicional dentro de seu corpo, chamada arguments, que é um objeto semelhante a um array que contém todos os valores passados para a função.

```
function adicionar() {  
    var soma = 0;  
    for (var i = 0, j = arguments.length; i < j; i++) {  
        soma += arguments[i];  
    }  
    return soma;  
}  
  
adicionar(2, 3, 4, 5); // 14
```

Funções

- Porém, isso não é tão útil. Vamos então criar uma função de média.

```
function media() {  
    var soma = 0;  
    for (var i = 0, j = arguments.length; i < j; i++) {  
        soma += arguments[i];  
    }  
    return soma / arguments.length;  
}  
  
media(2, 3, 4, 5); // 3.5
```

Funções

- Para reduzir um pouco a verbosidade desse código, podemos substituir o uso do array de argumentos por meio da Rest parameter syntax.

```
function media(...args) {  
    var sum = 0;  
    for (let valor of args) {  
        soma += valor;  
    }  
    return soma / args.length;  
}
```

```
media(2, 3, 4, 5); // 3.5
```

Funções

- E se quisermos encontrar a média de um array?

```
function mediaArray(arr) {  
    var soma = 0;  
    for (var i = 0, j = arr.length; i < j; i++) {  
        soma += arr[i];  
    }  
    return soma / arr.length;  
}
```

```
mediaArray([2, 3, 4, 5]); // 3.5
```

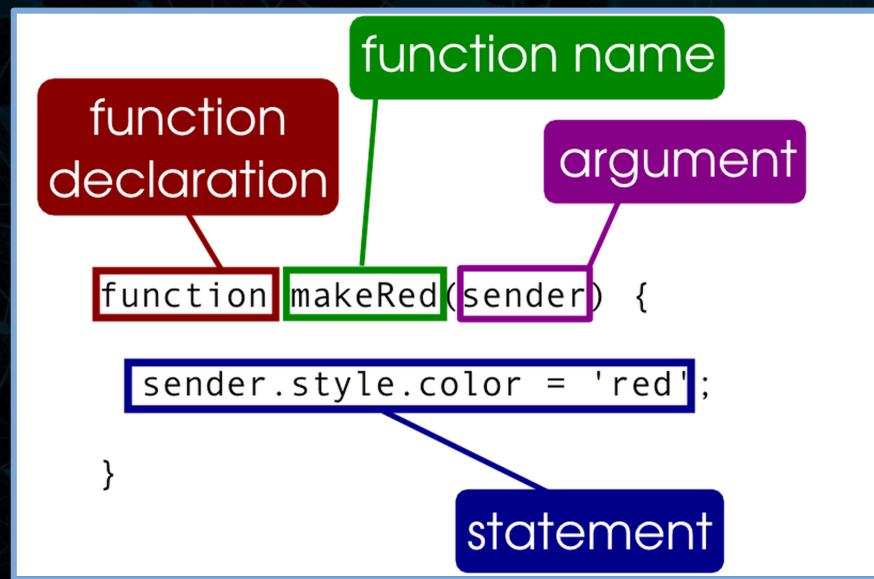
Funções

- JavaScript nos permite criar funções anônimas.

```
let media = function() {  
    var soma = 0;  
    for (var i = 0, j = arguments.length; i < j; i++) {  
        soma += arguments[i];  
    }  
    return soma / arguments.length;  
}  
  
console.log(media(2,3)); // 2.5
```

Funções

- Observe que as funções JavaScript são elas próprias objetos - como tudo mais em JavaScript - e você pode adicionar ou alterar propriedades nelas.



Inner Functions

- As declarações de função JavaScript são permitidas dentro de outras funções.
- Um detalhe importante das funções aninhadas em JavaScript é que elas podem acessar variáveis no escopo de sua função pai.

```
function f1() {  
    var a = 1;  
  
    function f2() {  
        var b = 4; // pai não pode usar esta variável  
        return a + b;  
    }  
    return f2(); // 5  
}
```

Closures

- Closures são uma das abstrações mais poderosas que o JavaScript tem a oferecer, mas também a mais potencialmente confusa.

```
function adder(a) {  
    return function(b) {  
        return a + b;  
    };  
}  
  
var add5 = adder(5);  
var add20 = adder(20);  
console.log(add5(6)); // 11  
console.log(add20(7)); // 27
```

Closures

- A função adder() cria novas funções 'adder', cada uma das quais, quando chamada com um argumento, adiciona-a ao argumento com o qual foi criada.
- O que está acontecendo aqui é praticamente o mesmo que estava acontecendo com as funções internas anteriormente: uma função definida dentro de outra função tem acesso às variáveis da função externa.

Closures

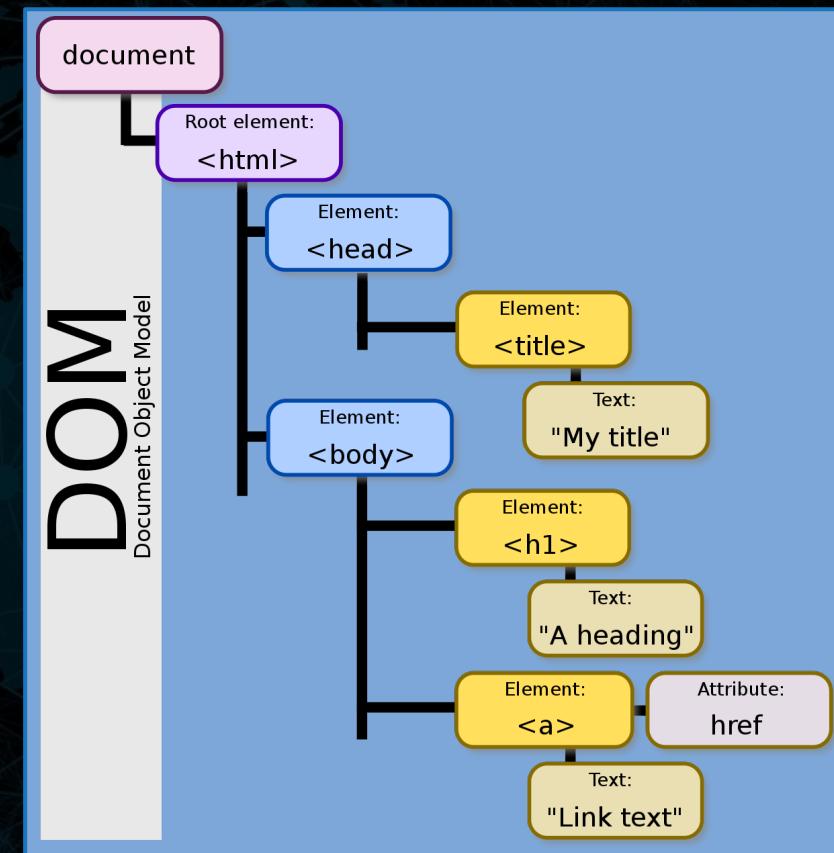
- A única diferença aqui é que a função externa retornou e, portanto, o bom senso parece ditar que suas variáveis locais não existem mais. Mas elas ainda existem - caso contrário, as funções do adder não funcionariam.
- Além do mais, existem duas "cópias" diferentes das variáveis locais de adder() - uma em que a é 5 e outra em que a é 20.

Closures

- O que está realmente acontecendo? Sempre que o JavaScript executa uma função, um objeto de 'escopo' é criado para conter as variáveis locais criadas dentro dessa função.
- Ele é inicializado com quaisquer variáveis passadas como parâmetros de função.
- É semelhante ao objeto global em que vivem todas as variáveis e funções globais, mas com algumas diferenças importantes: em primeiro lugar, um novo objeto de escopo é criado toda vez que uma função começa a ser executada e, em segundo lugar, ao contrário do objeto global (que é acessível como `this` e em navegadores como `window`), esses objetos de escopo não podem ser acessados diretamente de seu código JavaScript.

HTML DOM (Document Object Model)

- O HTML DOM é um modelo de objeto para HTML. Ele define:
 - Elementos HTML como objetos.
 - Propriedades para todos os elementos HTML.
 - Métodos para todos os elementos HTML.
 - Eventos para todos os elementos HTML.

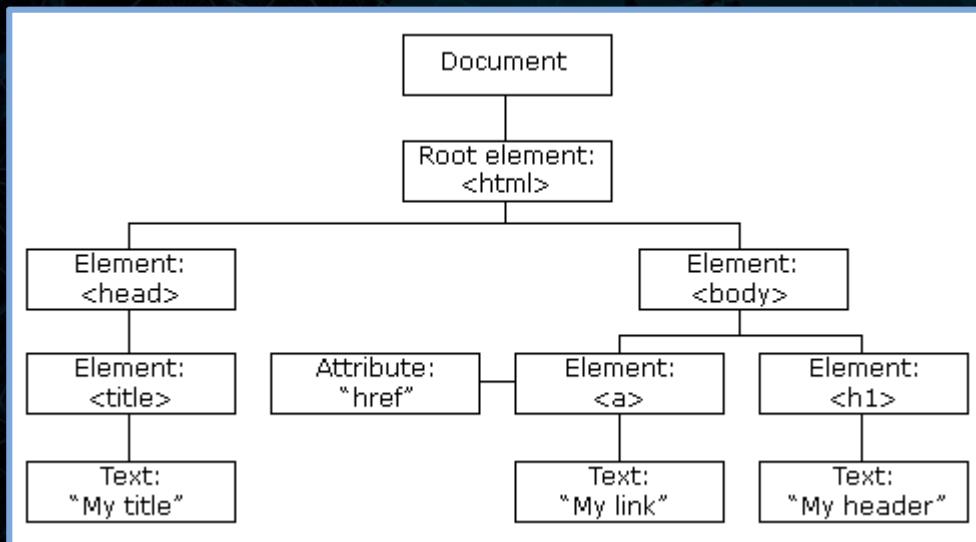


HTML DOM (Document Object Model)

- O HTML DOM é uma API (Application Programming Interface) para JavaScript:
 - JavaScript pode adicionar / alterar / remover elementos HTML.
 - JavaScript pode adicionar / alterar / remover atributos HTML.
 - JavaScript pode adicionar / alterar / remover estilos CSS.
 - JavaScript pode reagir a eventos HTML.
 - JavaScript pode adicionar / alterar / remover eventos HTML

HTML DOM (Document Object Model)

- Quando uma página da web é carregada, o navegador cria um Document Object Model da página.
- O modelo HTML DOM é construído como uma árvore de objetos:



Encontrando Elementos HTML

- Quando você deseja acessar elementos HTML com JavaScript, você deve encontrar os elementos primeiro.
- Há algumas maneiras de fazer isso:
 - Encontrar elementos HTML por id.
 - Encontrar elementos HTML por nome de tag.
 - Encontrar elementos HTML por nome de classe.
 - Encontrar elementos HTML por seletores CSS.
 - Encontrar elementos HTML por HTML object collections.

Encontrando Elemento por Id

- A maneira mais fácil de encontrar um elemento HTML no DOM é usando o id do elemento.
- Este exemplo encontra o elemento com id = "intro":

```
var elemento = document.getElementById("intro");
```

- Se o elemento for encontrado, o método retornará o elemento como um objeto (elemento), caso contrário, retornará null.

Encontrando Elemento por Tag

- Este exemplo encontra todos os elementos <p>:

```
var p = document.getElementsByTagName("p");
```

- Este exemplo encontra o elemento com id = "main" e, em seguida, encontra todos os elementos <p> dentro de "main":

```
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
```

Encontrando Elemento por Classe

- Se você deseja localizar todos os elementos HTML com o mesmo nome de classe, use `getElementsByClassName()`.
- Este exemplo retorna uma lista de todos os elementos com `class = "intro"`.

```
var x = document.getElementsByClassName("intro");
```

- Encontrar elementos por nome de classe não funciona no Internet Explorer 8 e em versões anteriores.

Encontrando Elemento por Seletor CSS

- Se você deseja encontrar todos os elementos HTML que correspondem a um seletor CSS especificado (id, nomes de classes, tipos, atributos, valores de atributos, etc), use o método querySelectorAll().
- Este exemplo retorna uma lista de todos os elementos <p> com class = "intro".

```
var x = document.querySelectorAll("p.intro");
```

Google APIs

- Google APIs são interfaces de programação de aplicativos (APIs) desenvolvidas pelo Google que permitem a comunicação com os serviços do Google e sua integração com outros serviços.
- Exemplos incluem:
 - Google Maps.
 - Google Fonts.
 - Google Charts.

XML

- XML significa eXtensible Markup Language.
- XML desempenha um papel importante em muitos sistemas de TI diferentes.
- XML é frequentemente usado para distribuir dados pela Internet.
- É importante que todos os desenvolvedores web tenham um bom entendimento de XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Gabriel</to>
  <from>Felippe</from>
  <heading>Lembrete</heading>
  <body>XML é importante!</body>
</note>
```

AJAX

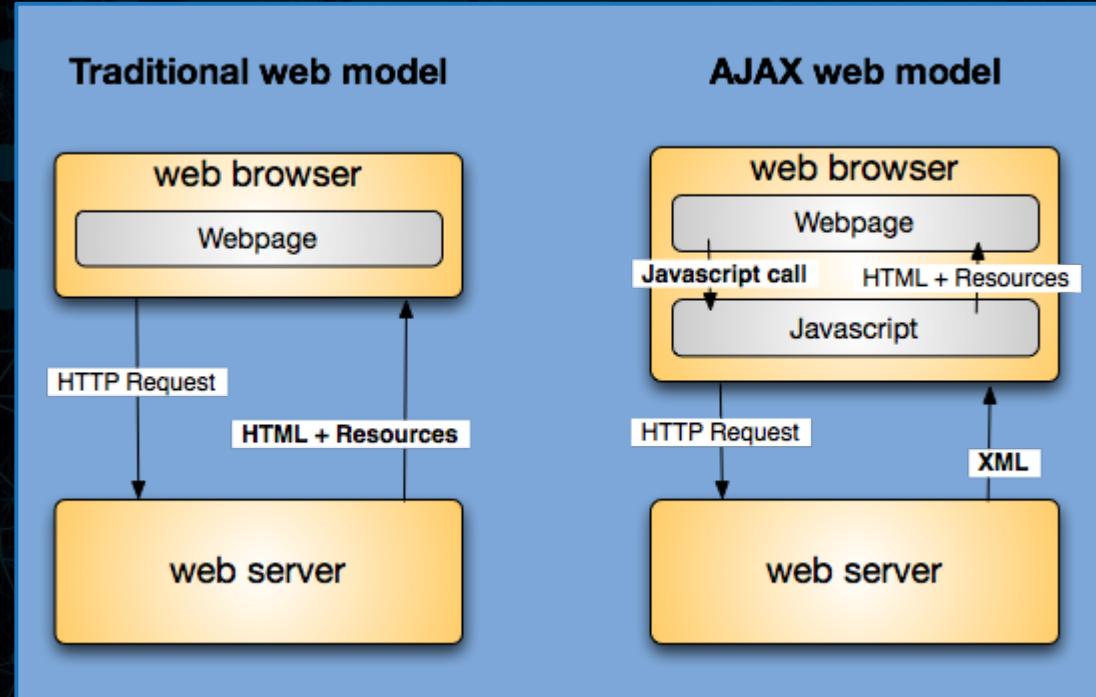
- AJAX é o sonho dos desenvolvedores, porque com ele podemos:
 - Ler dados de um servidor web - após o carregamento de uma página web.
 - Atualizar uma página web sem recarregar a página.
 - Enviar dados para um servidor web - em segundo plano.

AJAX

- **AJAX = Asynchronous JavaScript And XML.**
- **AJAX não é uma linguagem de programação.**
- **AJAX usa apenas uma combinação de:**
 - Um objeto XMLHttpRequest integrado ao navegador (para solicitar dados de um servidor web).
 - JavaScript e HTML DOM (para exibir ou usar os dados)
- **AJAX é um nome enganador. Os aplicativos AJAX podem usar XML para transportar dados, mas é igualmente comum transportar dados como texto simples ou texto JSON.**

AJAX

- O AJAX permite que as páginas web sejam atualizadas de forma assíncrona, trocando dados com um servidor da web nos bastidores. Isso significa que é possível atualizar partes de uma página web, sem recarregar a página inteira.



Como o AJAX Funciona

1. Um evento ocorre em uma página web (a página é carregada, um botão é clicado).
2. Um objeto XMLHttpRequest é criado por JavaScript.
3. O objeto XMLHttpRequest envia uma solicitação a um servidor web.
4. O servidor processa a requisição.
5. O servidor envia uma resposta de volta para a página web.
6. A resposta é lida pelo JavaScript.
7. Ação adequada (como atualização de página) é realizada pelo JavaScript.

JSON

- JSON significa JavaScript Object Notation.
- JSON é um formato leve para armazenar e transportar dados.
- JSON é frequentemente usado quando os dados são enviados de um servidor para uma página web.
- JSON é "autodescritivo" e fácil de entender!

Exemplo JSON

- Este exemplo define um objeto de pokémons:
Um array de 3 registros de pokémons (objetos)
é definido:

```
{  
  "pokemons": [  
    {"nome": "Pikachu", "tipo": "Elétrico"},  
    {"nome": "Charizard", "tipo": "Fogo"},  
    {"nome": "Abra", "tipo": "Psíquico"}  
  ]  
}
```

JSON Regras de Sintaxe

- Os dados estão em pares nome / valor.
- Os dados são separados por vírgulas.
- Chaves { } guardam os objetos.
- Os colchetes [] contêm arrays.
- O formato JSON é sintaticamente idêntico ao código para a criação de objetos JavaScript.
- Por causa dessa semelhança, um programa JavaScript pode facilmente converter dados JSON em objetos JavaScript nativos.

Convertendo JSON para Objeto JavaScript

- Um uso comum de JSON é ler dados de um servidor web e exibir os dados em uma página web.
- Para simplificar, isso pode ser demonstrado usando uma string como *input*.

```
var text = '{ "pessoas":[' +  
  '{ "nome":"Luís", "sobrenome":"Maria"},' +  
  '{ "nome":"José", "sobrenome":"Feliz"},' +  
  '{ "nome":"João", "sobrenome":"Batista" } ]}';
```

Convertendo JSON para Objeto JavaScript

- Em seguida, usamos a função incorporada do JavaScript `JSON.parse()` para converter a string em um objeto JavaScript:

```
var obj = JSON.parse(text);
```

- Por fim, usamos o novo objeto JavaScript na página web:

```
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
obj.pessoas[1].nome + " " + obj.pessoas[1].sobrenome;
</script>
```

Command Line Interface (CLI)

- CLI é um programa de linha de comando que aceita entrada de texto para executar funções do sistema operacional.
- Na década de 1960, usando apenas terminais de computador, essa era a única maneira de interagir com computadores.
- Nas décadas de 1970 e 1980, programas de linha de comando era comumente usada por sistemas Unix e sistemas de PC como MS-DOS e Apple DOS.
- A CLI ainda é usada por desenvolvedores de software e administradores de sistema para configurar computadores, instalar software e acessar recursos que não estão disponíveis na interface gráfica de usuário (GUI).

Command Line Interface (CLI)

- O gerenciador de pacotes de software npm usa a entrada de linha de comando para instalar softwares:

```
$ npm install -g gatsby-cli
```

- Podemos navegar em pastas (diretórios) com comandos de linha de comando:

```
$ cd documentos
```

```
$ cd ..
```

Comandos Linux Básicos

- ls - Lista o sistema de diretório (pasta).
- cd - Altera o diretório (pasta) no sistema de arquivos.
- cp - Copia um arquivo para outra pasta.
- mv - Move um arquivo para outra pasta.
- mkdir - Cria um novo diretório (pasta).
- rmdir - Remove um diretório (pasta).
- clear - Limpa a janela CLI.
- man command - Mostra o manual para um determinado comando.

Comandos Windows Básicos

- **dir** - Lista o sistema de diretório (pasta).
- **cd** - Altera o diretório (pasta) no sistema de arquivos.
- **copy** - Copia um arquivo para outra pasta.
- **move** - Move um arquivo para outra pasta.
- **mkdir** - Cria um novo diretório (pasta).
- **rmdir** - Remove um diretório (pasta).
- **cls** - Limpa a janela CLI.
- **help command** - Mostra o manual para um determinado comando.

NPM

- npm é a maior biblioteca de software do mundo (registro).
- npm também é um gerenciador e instalador de pacotes de software.
- Os desenvolvedores open-source usam o npm para compartilhar software.
- O npm é de uso gratuito.
- Você pode baixar todos os pacotes de software públicos npm sem qualquer registro ou logon.



NPM

- O npm inclui um Command Line Client que pode ser usado para baixar e instalar o software:

```
$ npm install <pacote>
```

- npm é instalado com Node.js.
- Isso significa que precisamos instalar o Node.js para que o npm seja instalado em nosso computador.

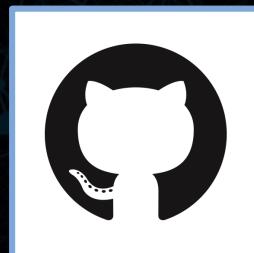
<https://nodejs.org/en/> 

Git

- Git é um software para rastrear alterações em qualquer conjunto de arquivos, geralmente usado para coordenar o trabalho entre programadores que desenvolvem código-fonte de forma colaborativa durante o desenvolvimento de software.
- Seus objetivos incluem velocidade, integridade de dados e suporte para fluxos de trabalho não lineares distribuídos (milhares de ramificações paralelas em execução em sistemas diferentes).
- Git foi criado por Linus Torvalds em 2005 para o desenvolvimento do kernel Linux, com outros desenvolvedores do kernel contribuindo para seu desenvolvimento inicial.

GitHub

- GitHub é uma plataforma de hospedagem de código para colaboração e controle de versão.
- O GitHub permite que você (e outros colegas) trabalhem juntos em projetos.
- Inscreva-se no GitHub em <https://github.com/>



GitHub

- Os fundamentos do GitHub são:

- Repositories
- Clone
- Branches
- Commits
- Pull Requests
- Git (O software de controle de versão no qual o GitHub é construído)

```
$ git push origin heroku  
$ git clone <repositório>  
$ cd /etc/  
$ ls
```

GitHub - Reppositório

- Um repositório GitHub pode ser usado para armazenar um projeto de desenvolvimento.
- Pode conter pastas e qualquer tipo de arquivo (HTML, CSS, JavaScript, Documentos, Dados, Imagens).
- Um repositório GitHub também deve incluir um arquivo de licença e um arquivo README sobre o projeto.
- Um repositório GitHub também pode ser usado para armazenar ideias ou quaisquer recursos que você deseja compartilhar.

GitHub - Branch

- Um branch do GitHub é usado para trabalhar com diferentes versões de um repositório ao mesmo tempo.
- Por padrão, um repositório tem um branch master (um branch de produção).
- Qualquer outro branch é uma cópia do branch master (como ele era em um determinado momento).
- Novos branches são para correções de bugs e trabalhos de recursos separados do master branch. Quando as alterações estiverem prontas, elas podem ser “merged” no branch master. Se você fizer alterações no branch master enquanto trabalha em um novo branch, essas atualizações podem ser obtidas.

GitHub - Pull Requests

- Pull Requests são o coração da colaboração do GitHub.
- Com um Pull Requests, você está propondo que suas alterações sejam merged (unidas) com o master.
- Pull Requests mostram diferenças de conteúdo, alterações, adições e subtrações em cores (verde e vermelho).

TypeScript



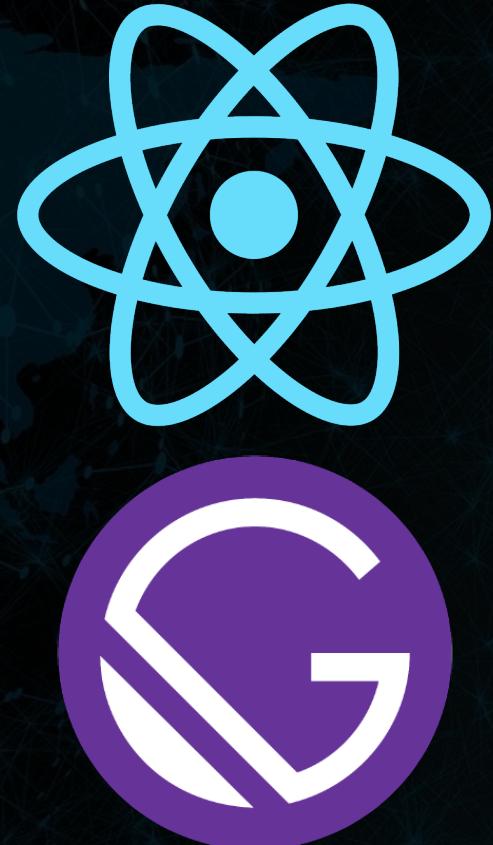
- TypeScript é uma linguagem de programação desenvolvida e mantida pela Microsoft.
- O TypeScript estende o JavaScript adicionando tipos.
- Tipos fornecem uma maneira de descrever a forma de um objeto, fornecendo melhor documentação e permitindo que o TypeScript valide se o seu código está funcionando corretamente.
- Os navegadores só podem executar JavaScript, então devemos transpilar TypeScript para JavaScript

Web Frameworks

- Um Web Framework (WF) é uma estrutura de software projetada para oferecer suporte ao desenvolvimento de aplicativos web, incluindo serviços web, recursos web e APIs web.
- Web Frameworks fornecem uma maneira padrão de construir e implantar aplicativos Web na World Wide Web.
- Os Web Frameworks visam automatizar a sobrecarga associada às atividades comuns realizadas no desenvolvimento web. Eles nos ajudam a desenvolver com eficiência e agilidade.

React

- React é uma biblioteca JavaScript de front end, de código aberto, para a construção de interfaces de usuário ou componentes de UI (User Interface).
- É mantido pelo Facebook e uma comunidade de desenvolvedores individuais e empresas.
- O React pode ser usado como base no desenvolvimento de aplicativos single-page ou mobile.



Vue.js

- O Vue.js (comumente conhecido como Vue, pronunciado "view") é um framework progressivo do JavaScript de código aberto (open source) para a construção de interfaces de usuário.
- A integração em projetos que usam outras bibliotecas de JavaScript é facilitada com o Vue porque ele foi projetado para ser adotado de forma incremental.
- Vue.js nos permite estender o HTML com atributos HTML chamados directives.



Django

- Django é uma web framework Python de alto nível que incentiva o desenvolvimento rápido e um design limpo e pragmático.
- Construído por desenvolvedores experientes, ele cuida de grande parte do trabalho de desenvolvimento Web, para que possamos nos concentrar em escrever nosso aplicativo sem precisar reinventar a roda. É gratuito e de código aberto.

django

Laravel

- Laravel é um framework de aplicação web PHP com sintaxe expressiva e elegante.
- Criado por Taylor Otwell e destinado ao desenvolvimento de aplicações web seguindo o padrão arquitetônico model – view – controller.



Full Stack Web Developer

- Um Desenvolvedor Full Stack é uma pessoa que pode desenvolver software cliente e servidor.
- Além de dominar HTML e CSS, ele também sabe tecnologias como:
 - Programar um navegador / browser (como usar JavaScript, jQuery, Angular ou Vue).
 - Programar um servidor (como usar PHP, ASP, Python ou Node).
 - Programar um banco de dados (como usar SQL, SQLite ou MongoDB).

Client Software (Front End)

- HTML
- CSS
- Bootstrap
- W3.CSS
- JavaScript
- ES5
- HTML DOM
- JSON
- XML
- jQuery
- Angular
- React
- Backbone.js
- Ember.js
- Redux
- Storybook
- GraphQL
- Meteor.js
- Grunt
- Gulp

Server Software (Back End)

- PHP
- ASP
- C++
- C#
- Java
- Python
- Node.js
- Express.js
- Ruby
- REST
- GO
- SQL
- MongoDB
- Firebase.com
- Sass
- Less
- Parse.com
- PaaS (Azure and Heroku)

Stacks Populares

- LAMP stack: JavaScript - Linux - Apache - MySQL - PHP.
- LEMP stack: JavaScript - Linux - Nginx - MySQL - PHP.
- MEAN stack: JavaScript - MongoDB - Express - Angular - Node.js.
- Django stack: JavaScript - Python - Django - MySQL.
- Ruby on Rails: JavaScript - Ruby - SQLite - Rails

Deployment

- Deployment em software e desenvolvimento web significa enviar mudanças ou atualizações de um ambiente de implantação para outro.
- Ao configurar um site, você sempre terá seu site ativo, que é chamado de ambiente ativo ou ambiente de produção.
- Se desejarmos a capacidade de fazer alterações sem que elas afetem nosso site ativo, podemos adicionar ambientes adicionais. Esses ambientes são chamados de ambientes de desenvolvimento ou ambientes de implantação.

Deployment

- O mais comum modelo de implantação é o clássico da “left to right” ao trabalhar com vários ambientes de implantação.
- Neste modelo, as alterações são feitas em ambientes locais, de desenvolvimento ou de teste (dependendo da configuração) e enviadas da esquerda para a direita através dos diferentes ambientes que terminam no ambiente ao vivo. Assim que o processo de implantação for concluído, as novas alterações ficarão visíveis no ambiente ativo.



Deployment

- Existem empresas de computação em nuvem que oferecem hospedagens e serviços de back-end *serverless* para aplicativos da web e sites estáticos.



heroku



SQL

- SQL significa Structured Query Language.
- SQL é uma linguagem padrão para acessar bancos de dados.
- SQL é um padrão internacional (ISO) desde 1987.
- Para acessar um banco de dados, você usa instruções SQL.
- A seguinte instrução SQL seleciona todos os registros em uma tabela de banco de dados chamada "Clientes":

```
SELECT * FROM Clientes;
```

Tabelas de Banco de Dados

- Um banco de dados geralmente contém uma ou mais tabelas.
- Cada tabela é identificada por um nome como "Clientes" ou "Pessoas".
- Abaixo temos uma seleção de uma tabela "Clientes":

ID	Nome	Cidade	País
1	José Paulo	São Paulo	Brasil
2	Pedro Luís	Salvador	Brasil
3	João Carlos	Porto Alegre	Brasil
4	Lucas Jesus	Joinville	Brasil

Tabelas de Banco de Dados

- A tabela anterior contém quatro registros (um para cada cliente) e quatro colunas:
 - ID do Cliente.
 - Nome do Cliente.
 - Cidade do Cliente.
 - País do Cliente.

Instruções SQL Importantes

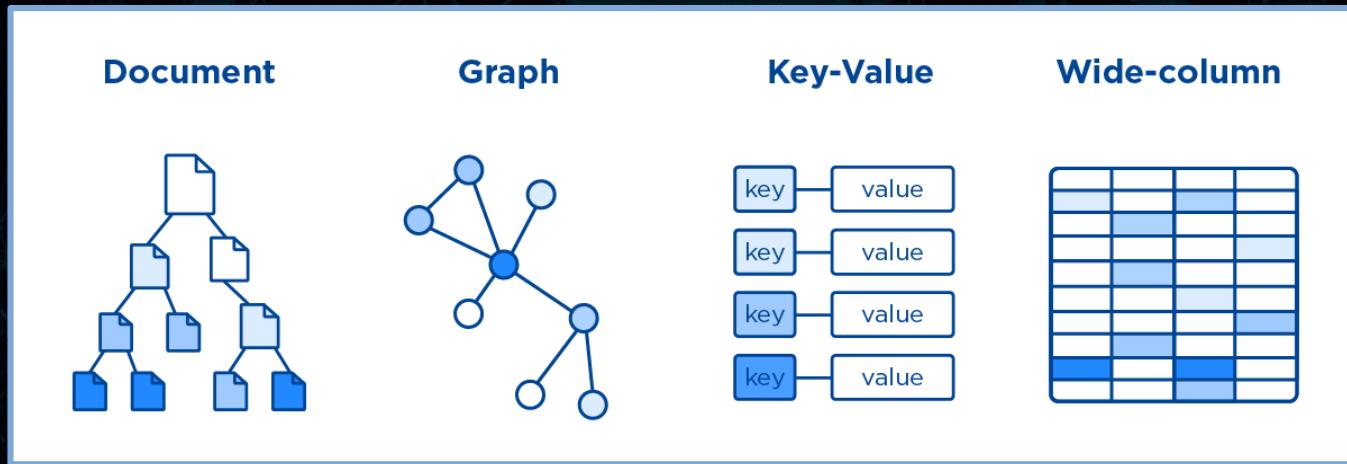
- SELECT - extrai dados de um banco de dados.
- UPDATE - atualiza dados em um banco de dados.
- DELETE - exclui dados de um banco de dados.
- INSERT INTO - insere novos dados em um banco de dados.
- CREATE DATABASE - cria um novo banco de dados

Instruções SQL Importantes

- ALTER DATABASE - modifica um banco de dados.
- CREATE TABLE - cria uma nova tabela.
- ALTER TABLE - modifica uma tabela.
- DROP TABLE - exclui uma tabela.
- CREATE INDEX - cria um índice (chave de pesquisa).
- DROP INDEX - exclui um índice.

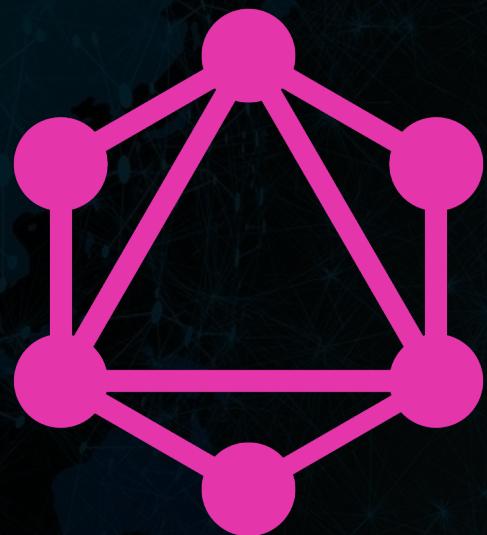
NoSQL

- Um banco de dados NoSQL (originalmente referindo-se a "não SQL" ou "não relacional") fornece um mecanismo para armazenamento e recuperação de dados que são modelados em meios diferentes das relações tabulares usadas em bancos de dados relacionais.



GraphQL

- GraphQL é uma linguagem de consulta e manipulação de dados de código aberto para APIs e um runtime para atender consultas com dados existentes.
- Ele fornece uma abordagem para desenvolver APIs web e foi comparado e contrastado com REST e outras arquiteturas de serviço web.



Web Performance

- Web Performance se refere à velocidade com que as páginas da web são baixadas e exibidas no navegador do usuário.
- Velocidades mais rápidas de download de sites têm demonstrado aumentar a retenção e a fidelidade do visitante e a satisfação do usuário, especialmente para usuários com conexões lentas à Internet e aqueles em dispositivos móveis.
- Alguns aspectos que podem afetar a velocidade de carregamento da página web incluem cache do navegador / servidor, otimização de imagem e criptografia (por exemplo, SSL), que pode afetar o tempo que leva para as páginas serem renderizadas.

Ferramentas para Medir Performance

- Existem ferramentas capazes de relatar o desempenho de uma página em dispositivos móveis e desktop e fornece sugestões sobre como essa página pode ser aprimorada para otimização.

<https://developers.google.com/speed/pagespeed/insights/>

<https://gtmetrix.com/>

<https://tools.pingdom.com/>



Concluindo

- Desenvolvimento Web é divertido e poderoso.
- Nos permite exercer a nossa criatividade.
- Existem um grande número de tecnologias que podem nos auxiliar no processo de Desenvolvimento Web.
- Existem diferentes níveis de complexidades de aplicações Web.

Referências

- Computer Hope: <https://www.computerhope.com>
- Stanford CS193X:
<https://web.stanford.edu/class/archive/cs/cs193x/cs193x.1176/>
- Developer Mozilla:
<https://developer.mozilla.org/en-US/docs/Learn>
- W3Schools: <https://www.w3schools.com/>

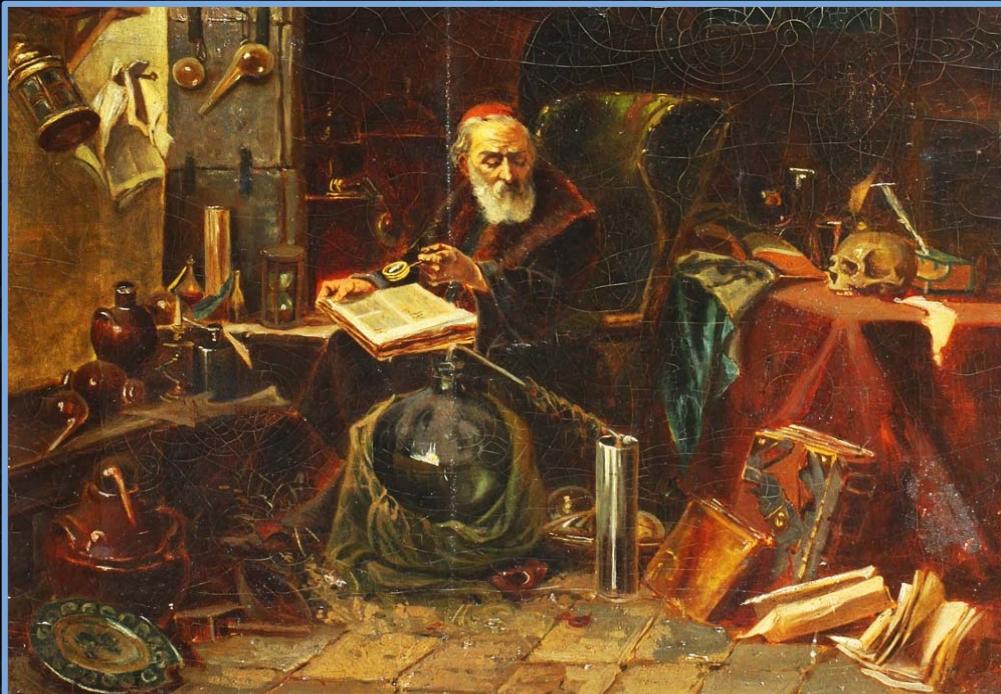
“Do the difficult things while they are easy and do the great things while they are small. A journey of a thousand miles must begin with a single step.”

Lao Tzu



Caminho do Conhecimento

<https://github.com/kamranahmedse/developer-roadmap>



Bons Estudos!