

HTML, CSS e JavaScript

SEL 0373 - Projeto em Internet das Coisas

Alunos:

Gianlucca Otuski Belluomini - 13676624

Luigi Yassuo Nakata - 13838325

Orientador:

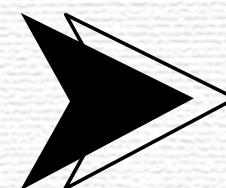
Prof. Dr. José Roberto Boffino de Almeida Monteiro

HTML - História



1991

A primeira versão do HTML foi publicada em 1991 por Tim Berners-Lee, o criador da World Wide Web. Ele publicou a primeira versão do HTML em 1991, consistindo em 18 tags.



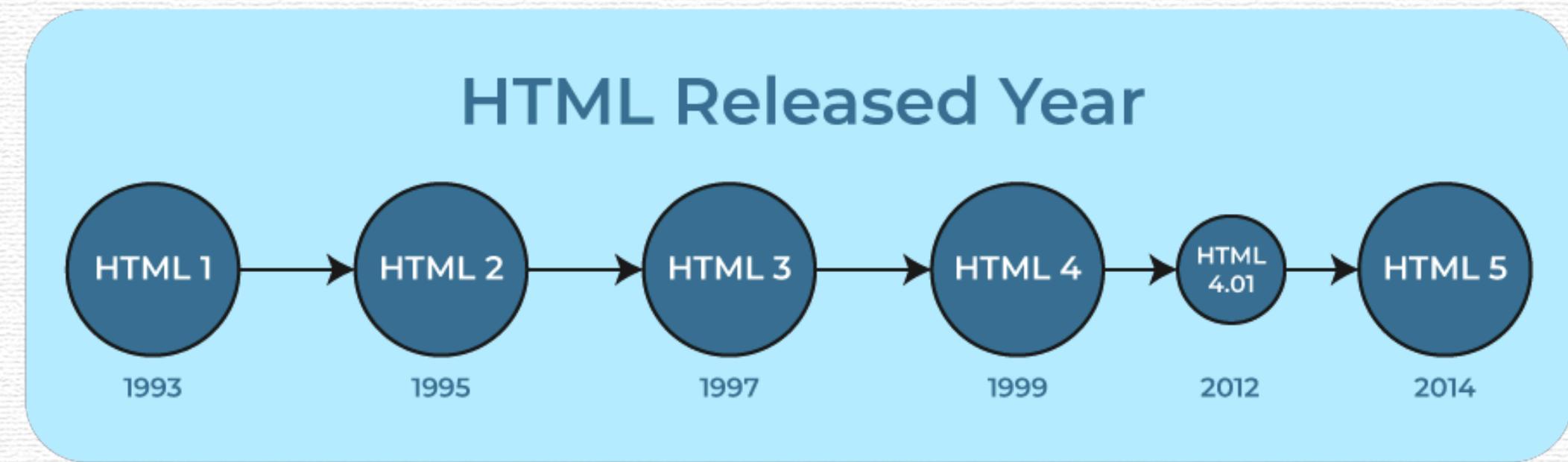
1993

A primeira versão formal da linguagem. Criada em 1991 e documentada como um rascunho em 1993, foi um marco inicial para a Web. Embora tenha estabelecido as bases do HTML, não chegou a ser padronizada oficialmente pelo IETF.



2014

O maior upgrade da linguagem foi o lançamento do HTML5 em 2014. Diversas novas tags semânticas foram adicionadas que revelam o significado do seu próprio conteúdo, como <article>, <header>, e <footer>.



HTML - Introdução

- HTML ou "Linguagem de Marcação de Hipertexto" é uma linguagem de marcação utilizada na construção de páginas na Web.
- Uma linguagem de marcação é um tipo de linguagem que utiliza marcas ou tags (como `<html>`, `<head>`, `<p>`, etc) para estruturar, organizar e formatar o conteúdo de um documento.
- O HTML não é considerado uma linguagem de programação, já que ele não pode criar funcionalidades dinâmicas.



HTML - Exemplos de Usos

- Criação de sites: Os desenvolvedores utilizam HTML para definir a estrutura e a apresentação dos elementos em um navegador, como textos, links e mídias.
- Navegação na internet: Como o HTML permite a inserção de hiperlinks, os usuários podem acessar diferentes páginas e sites de forma intuitiva e interconectada.

Exemplo

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

HTML - Funcionamento

- Um site médio inclui diversas páginas HTML diferentes. Cada uma delas possui um arquivo HTML separado.
- Documentos HTML são arquivos que terminam com uma extensão .html ou .htm. Um navegador lê o arquivo HTML e renderiza o seu conteúdo para que os usuários da internet possam vê-lo.
- Todas as páginas HTML possuem uma série de elementos, que consistem num conjunto de tags e atributos. Uma tag diz para o navegador onde um elemento começa e termina, enquanto um atributo descreve as características de um elemento.



HTML - Funcionamento

As três principais partes de um elemento são:

1 - Tag de abertura

A tag de abertura é usada para dizer onde um elemento começa a ter efeito. A tag é cercada de colchetes angulares para abertura e fechamento. Por exemplo, a tag de abertura `<p>` serve para criar um parágrafo.

2 - Conteúdo

Conteúdo é a parte que os usuários verão.

3 - Tag de fechamento

A tag de fechamento é igual à tag de abertura, mas com uma barra antes do nome do elemento. Por exemplo, `</p>` para encerrar um parágrafo.

`<p> É assim que você adiciona um parágrafo no HTML. </p>`

HTML - Funcionamento

A maioria dos elementos possui uma tag de abertura e de fechamento, mas alguns não precisam fechar a tag para funcionar. Esse é o caso dos elementos vazios. Eles não usam uma tag de fechamento pois não têm conteúdo:

```

```

Essa tag de imagem possui dois atributos — um atributo `src` (que é um caminho de imagem) e um atributo `alt` (que é o texto de descrição). Contudo, ele não tem conteúdo nem uma tag de fechamento.

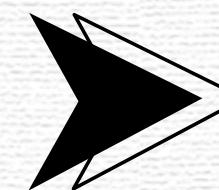
Por fim, cada documento HTML deve começar com uma declaração `<!DOCTYPE>` para informar ao navegador qual é o tipo de documento. Com o HTML5, a declaração doctype HTML pública será:

```
<!DOCTYPE html>
```

HTML - Funcionamento

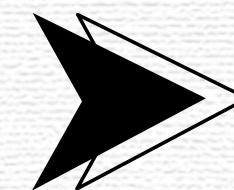
```
1  <!DOCTYPE html>
2  <html lang=""pt-BR">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>_título_</title>
7  </head>
8  <body>
9
10     <h1>_cabeçalho_</h1>
11     <p>Clique no link abaixo para acessar o Google:</p>
12
13     <a href="https://www.google.com" target="_blank">Ir para o Google</a>
14 </body>
15 </html>
```

CSS - História



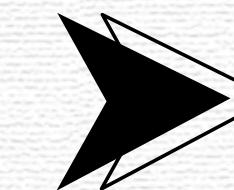
1994

O conceito do CSS foi proposto por Håkon Wium Lie, um cientista da computação norueguês que trabalhava no CERN junto com Tim Berners-Lee. A ideia era permitir que os desenvolvedores aplicassem estilos a documentos HTML sem misturar estrutura e design.



1996

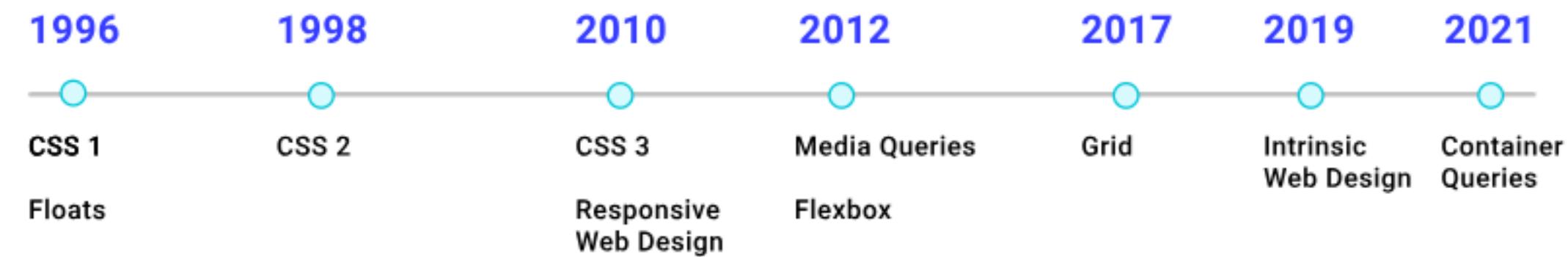
A primeira versão do CSS foi lançada pela W3C, permitindo estilos básicos como fontes, cores e espaçamentos. Porém, o suporte nos navegadores era limitado.



1998

Com essa atualização, o CSS passou a oferecer suporte a posicionamento de elementos (como absolute e relative), classes para formatação mais específica e o conceito de media types (como estilos específicos para impressão).

CSS layout over time



CSS - Introdução

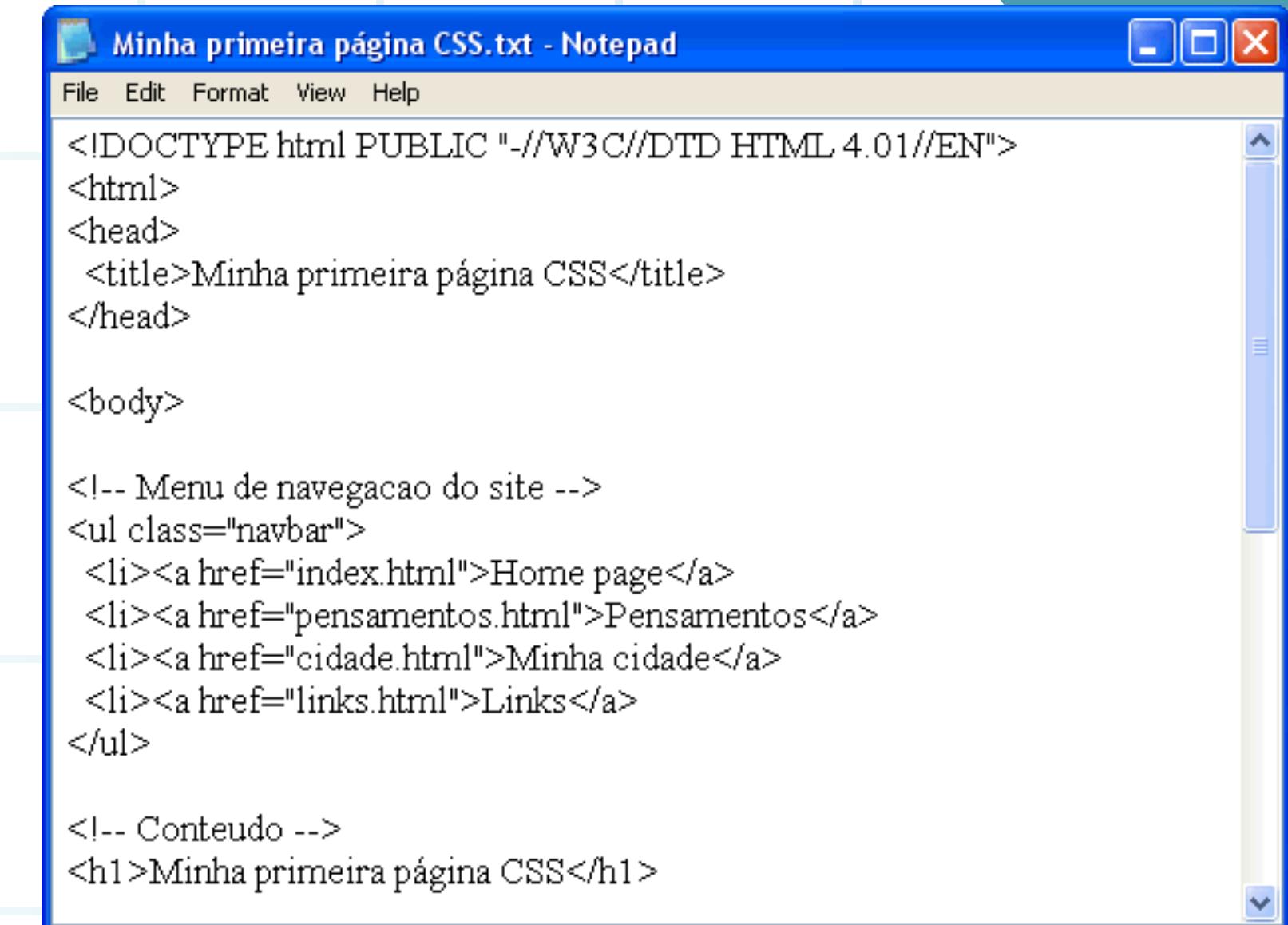
- CSS ou "Cascading Style Sheets" é uma linguagem de estilo utilizada na formatação e aparência de páginas na Web.
- Essa linguagem define como os elementos de uma página HTML devem ser apresentados, controlando aspectos visuais como cores, fontes, tamanhos e layouts.
- CSS não é considerado uma linguagem de programação, já que não possui a capacidade de criar lógicas ou funcionalidades dinâmicas. Ele é uma linguagem de estilo responsável apenas pela aparência visual das páginas.

CSS



CSS - Exemplos de Usos

- Estilização de sites: O CSS permite que desenvolvedores personalizem cores, fontes, tamanhos e espaçamentos, tornando as páginas mais atraentes e organizadas.
- Layouts responsivos: Com técnicas como Flexbox e Grid, o CSS possibilita que sites se adaptem a diferentes tamanhos de tela, garantindo uma boa experiência em dispositivos móveis e desktops.
- Modo escuro e acessibilidade: CSS permite que sites ofereçam temas claros e escuros, além de melhorar a acessibilidade ajustando contrastes, tamanhos de fonte e espaçamentos para facilitar a leitura.



The screenshot shows a Windows Notepad window with a blue title bar and a white body. The title bar says "Minha primeira página CSS.txt - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main content area contains the following HTML code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
    <title>Minha primeira página CSS</title>
</head>

<body>

    <!-- Menu de navegacao do site -->
    <ul class="navbar">
        <li><a href="index.html">Home page</a>
        <li><a href="pensamentos.html">Pensamentos</a>
        <li><a href="cidade.html">Minha cidade</a>
        <li><a href="links.html">Links</a>
    </ul>

    <!-- Conteudo -->
    <h1>Minha primeira página CSS</h1>
```

CSS - Funcionamento

O CSS é utilizado para definir a aparência visual dos elementos em um documento HTML. Ele pode ser inserido de três formas:

- Inline: Diretamente no próprio elemento HTML
- Interno: Dentro da tag `<style>` no cabeçalho do documento HTML.
- Externo: Em um arquivo CSS separado, vinculado ao documento HTML.

```
1  <!-- CSS Interno -->
2  <style>
3  |   h1 { color: blue; font-size: 30px; }
4  </style>
5
6  <!-- CSS Externo -->
7  <link rel="stylesheet" href="styles.css">
8  </head>
9  <body>
10
11 <!-- CSS Inline -->
12 <h1 style="color: red; font-size: 40px;">Título com CSS Inline</h1>
```

CSS - Funcionamento

Estrutura Básica de uma Regra CSS

- 1 - Seletor:
 - Indica qual elemento HTML será estilizado.
 - Exemplo: p, h1, .classe, #id.
- 2 - Propriedade:
 - Define o estilo a ser aplicado ao elemento.
 - Exemplo: color, font-size, margin.
- 3 - Valor:
 - Especifica o valor para a propriedade.
 - Exemplo: red, 16px, 10px.

```
1 < .classe {  
2     color: red;  
3     font-size: 25px;  
4     font-weight: bold;  
5     background: #26df23;  
6 }
```

CSS - Funcionamento

Tipos de Seletores no CSS

- Seletor de Elemento: Seleciona todos os elementos de um tipo específico.
 - Exemplo: `h1 { color: red; }` (todos os `<h1>` terão a cor vermelha)
- Seletor de Classe: Seleciona elementos com uma classe específica.
 - Exemplo: `.button { background-color: blue; }` (todos os elementos com a classe button terão fundo azul)
- Seletor de ID: Seleciona um único elemento com um ID específico.
 - Exemplo: `#header { font-size: 24px; }` (o elemento com ID header terá a fonte 24px)

CSS - Funcionamento

Cascata

O termo "cascata" se refere à forma como o CSS aplica estilos de acordo com a especificidade e a ordem de aplicação. Quando múltiplas regras podem ser aplicadas a um mesmo elemento, o CSS segue uma hierarquia:

- Estilos internos (em uma tag `<style>` dentro do HTML) têm maior prioridade do que os externos.
- ID tem maior especificidade do que classe, e classe tem maior prioridade do que elemento.
- Caso haja conflito de regras com a mesma especificidade, a última regra aplicada tem a maior prioridade.

CSS - Funcionamento

O Box Model define como o conteúdo de um elemento HTML é exibido e organizado. Ele é composto por:

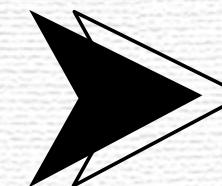
- Content: O conteúdo real do elemento (texto, imagens).
- Padding: Espaço entre o conteúdo e a borda.
- Border: Borda que envolve o elemento.
- Margin: Espaço fora da borda do elemento.

The CSS Box Model

The CSS box model is essentially a box that wraps around every HTML element.

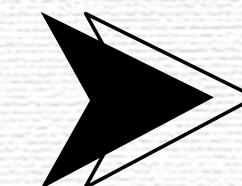
```
div {  
    width: 200px;  
    height: 200px;  
    border: 2px solid grey;  
    padding: 20px;  
    margin: 20px;  
}
```

JavaScript - História



1995

O JavaScript foi criado por Brendan Eich, um engenheiro da Netscape Communications, em 1995. Originalmente chamado de Mocha, depois foi renomeado para LiveScript e, por fim, para JavaScript.



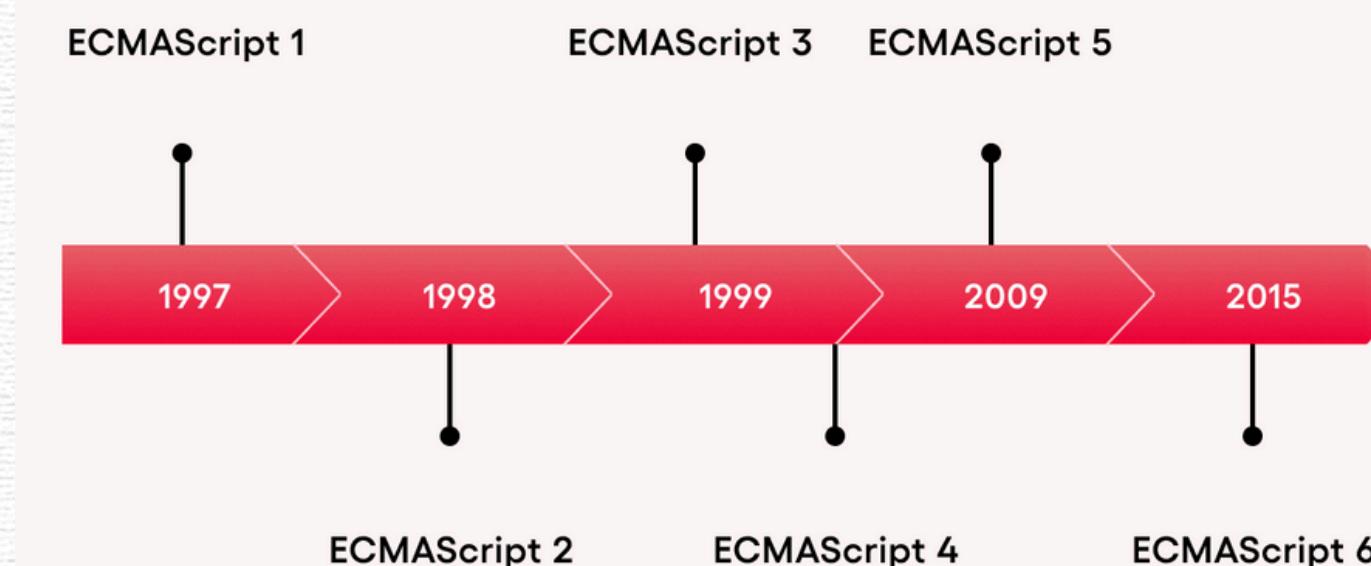
1997

O JavaScript passou a ser padronizado pela ECMA International, uma organização de padronização, com o nome de ECMAScript. Isso foi feito para garantir que a linguagem tivesse um desenvolvimento consistente e pudesse ser implementada em diferentes navegadores.



2015

O maior upgrade da linguagem foi o lançamento do ECMAScript 6 (ES6) em 2015. Diversos novos recursos foram adicionados, como as palavras-chave `let` e `const` para declaração de variáveis, funções de seta para uma sintaxe mais concisa, classes para programação orientada a objetos e Promises para facilitar o trabalho com código assíncrono.



JavaScript- Introdução

- O JavaScript é uma linguagem de programação amplamente utilizada para criar páginas web interativas e dinâmicas.
- Ele é executado no lado do cliente (navegador) e permite manipular elementos HTML, realizar requisições assíncronas (como chamadas de API), e criar efeitos interativos sem a necessidade de recarregar a página.
- O JavaScript é essencial para a construção de aplicações web modernas, sendo frequentemente combinado com HTML e CSS.



JavaScript- Exemplos de Usos

- Criação de sites: O JavaScript permite adicionar interatividade a sites criados com HTML. Por exemplo, ao clicar em um botão, é possível mostrar ou esconder elementos na página, validar formulários ou até mesmo realizar cálculos diretamente no navegador. Isso torna o site mais dinâmico e envolvente.
- Navegação na internet: JavaScript é frequentemente utilizado para melhorar a experiência de navegação. Por exemplo, ele pode ser usado para atualizar o conteúdo de uma página sem recarregar toda a página (técnica conhecida como AJAX), ou para criar menus de navegação interativos que respondem a cliques ou movimentos do mouse, proporcionando uma navegação mais fluida.

```
● ● ●  
1 const Pessoa = {  
2   nome: "Pedro",  
3   sobrenome: "Souza",  
4  
5   obterInformacao: function() {  
6     console.log(this.nome, this.sobrenome);  
7   },  
8 };  
9  
10 Pessoa.obterInformacao(); // Pedro Souza  
11
```

JavaScript - Funcionamento

1. Execução no Navegador:

- Quando um navegador carrega uma página web, ele carrega o código HTML, CSS e JavaScript.
- O JavaScript é executado pelo motor JavaScript do navegador (como o V8 no Chrome, SpiderMonkey no Firefox, ou Chakra no Edge).
- Esse motor lê e interpreta o código JavaScript em tempo real, o que permite a interação com o DOM (Document Object Model), tornando a página dinâmica.

```
1 // 1. Execução no Navegador - Manipulação do DOM
2
3 // Esse código modifica o texto desse elemento dinamicamente.
4 document.getElementById('texto_inicial').innerText = 'texto_final';
```

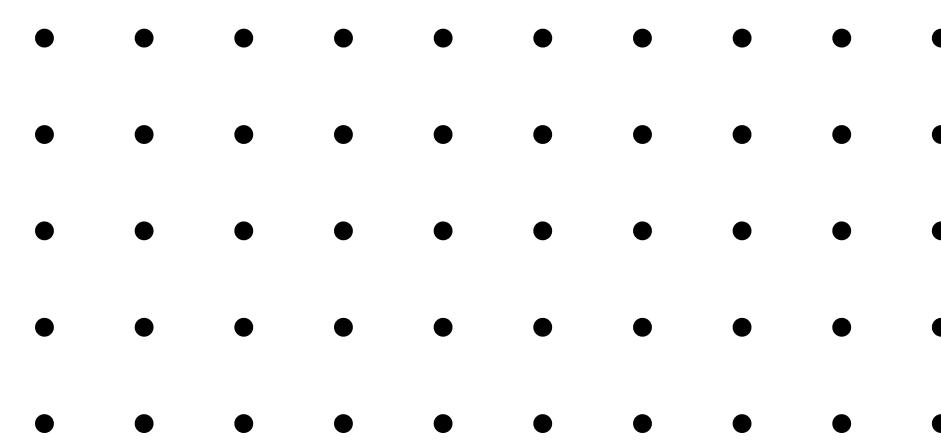
JavaScript - Funcionamento

2. Event Loop (Laço de Eventos):

- O JavaScript é uma linguagem single-threaded (de execução única), o que significa que ele executa uma tarefa por vez. Para lidar com múltiplas operações sem travar o navegador, o JavaScript utiliza um Event Loop.
- O Event Loop trabalha com duas filas principais: a Call Stack (pilha de chamadas) e a Task Queue (fila de tarefas).
- Call Stack: Onde o JavaScript executa funções.
- Task Queue: Onde as funções aguardam para serem executadas depois que a Call Stack estiver vazia.
- A ideia é que, quando há operações assíncronas (como timers, requisições de rede, ou eventos de UI), elas são colocadas na Task Queue e são executadas somente quando a Call Stack estiver livre.

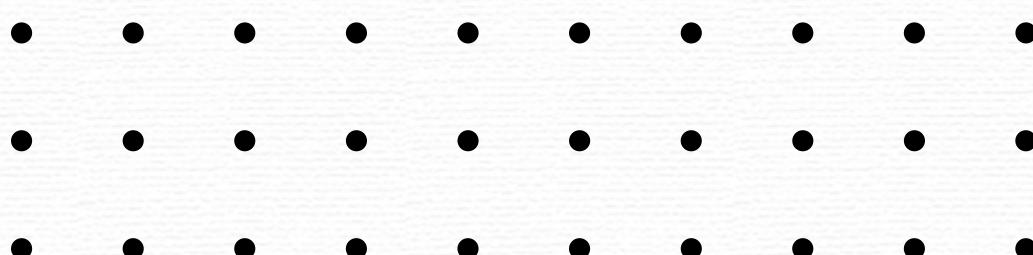
```
1 // 2. Event Loop - Como funciona a execução assíncrona no JavaScript
2
3 console.log('Início');
4
5 // A função setTimeout é uma operação assíncrona. Ela coloca o código para ser executado
6 // depois de um intervalo de tempo, sem bloquear a execução do código seguinte.
7 ↘ setTimeout(() => {
8   | console.log('Tarefa assíncrona executada após 1 segundo');
9   }, 1000);
10
11 console.log('Fim');
```

JavaScript - Funcionamento



3. Funções Assíncronas e Promises:

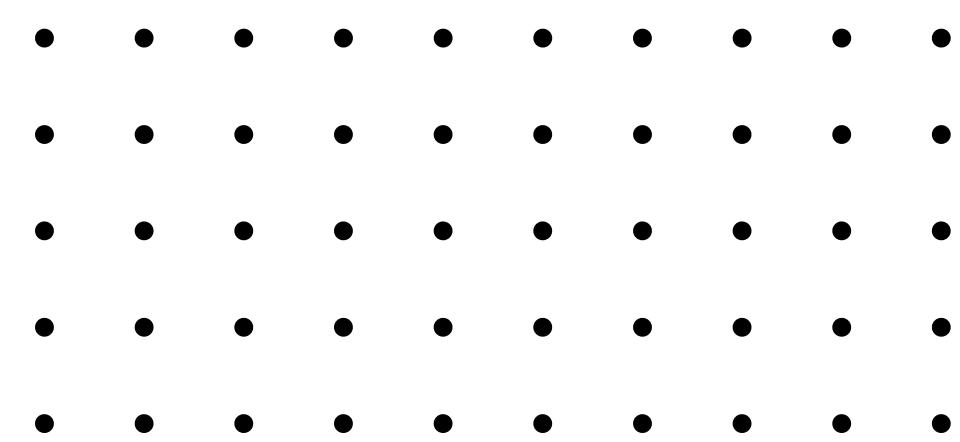
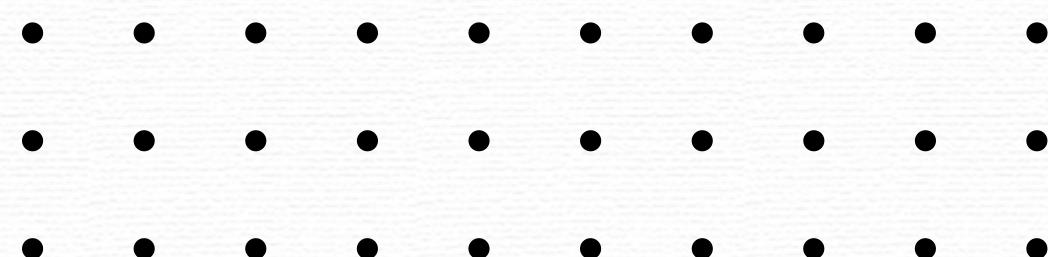
- JavaScript lida com operações assíncronas (como carregar dados de uma API) usando callbacks, Promises e async/await.
- Callbacks: São funções passadas como argumento para outras funções, que são executadas depois que a operação assíncrona é concluída.
- Promises: Representam um valor que pode estar disponível agora ou em algum momento no futuro. Elas podem estar em três estados: pendente, resolvida ou rejeitada.
- async/await: Sintaxe que facilita a escrita de código assíncrono de maneira mais legível, tratando Promises como se fossem funções síncronas.



JavaScript - Funcionamento

4. Escopo e Closures:

- O escopo determina a visibilidade das variáveis e funções em diferentes partes do código.
- Escopo global: Variáveis e funções acessíveis em qualquer lugar no código.
- Escopo local: Variáveis e funções acessíveis apenas dentro de uma função ou bloco.
- Closures ocorrem quando uma função lembra do seu ambiente de execução, mesmo após a execução de seu código ter terminado. Isso é útil para criar funções privadas e gerenciar estados.



JavaScript - Funcionamento

5. Objetos e Prototypes:

- Objetos em JavaScript são coleções de pares chave-valor. Eles são fundamentais para a estrutura da linguagem e para a manipulação de dados.
- Prototype: Cada objeto em JavaScript possui um protótipo, e isso cria uma cadeia de herança chamada prototype chain, permitindo que objetos herdem propriedades e métodos de outros objetos.

```
1  ↴ const pessoa = {  
2      nome: 'João',  
3      sobrenome: 'Silva',  
4  ↴     nomeCompleto: function() {  
5      return this.nome + ' ' + this.sobrenome;  
6    }  
7  };  
8  
9  // Criação do objeto funcionario, que herda de pessoa  
10 const funcionario = Object.create(pessoa);  
11 |  
12 // Inicialmente, funcionario não tem as propriedades  
13 // nome e sobrenome, então ele herda de pessoa.  
14 console.log(funcionario.nome); // João  
15 console.log(funcionario.sobrenome); // Silva
```

JavaScript - Funcionamento

6. Modularização:

- O JavaScript moderno permite que você divida seu código em módulos, usando import e export. Isso facilita a manutenção e a reutilização de código.

```
1 // No arquivo math.js (módulo)
2 export function sum(a, b) {
3   return a + b;
4 }
5
6 // No arquivo main.js
7 import { sum } from './math.js'; // Importa a função "sum" do módulo math.js
8
9 console.log(sum(3, 5)); // 8
```