Aula Introdutória de JavaScript para Uso com HTML

Karan Luciano

Curso de Análise e Desenvolvimento de Sistemas

1 de outubro de 2024

Sumário

1. Introdução ao JavaScript
2. Como Incluir JavaScript em HTML
3. Plugins - VsCode
4. Exemplos Básicos
5. Estrutura do jQuery
6. Conclusão

O que é JavaScript?

- Linguagem de programação de alto nível e interpretada.
- Utilizada para adicionar **interatividade** às páginas web.
- Complementa HTML e CSS, manipulando o conteúdo e o comportamento da página em tempo real.

História do JavaScript

- Criado em 1995 por Brendan Eich na Netscape.
- Inicialmente chamado de Mocha, depois LiveScript, e finalmente JavaScript.
- O nome foi escolhido por motivos de marketing, associando-se ao sucesso do Java.

Curiosidade: Apesar do nome, JavaScript e Java são linguagens distintas.

Conceito de Client-side no JavaScript

- **Client-side** refere-se à execução de scripts no navegador do usuário.
- No JavaScript, o código é processado diretamente no navegador, tornando as páginas web dinâmicas e interativas.
- Exemplos de uso incluem validação de formulários, manipulação do DOM e animações.
- Não requer comunicação com o servidor para executar, o que resulta em respostas mais rápidas.

Nota: O JavaScript *Client-side* é fundamental para melhorar a experiência do usuário.

O que é ECMAScript?

- ECMAScript é um padrão de linguagem de script especificado pela ECMA International.
- Base para JavaScript, JScript (Microsoft) e ActionScript (Adobe).
- Define como a linguagem deve ser implementada pelos navegadores e outras plataformas.
- Permite a interoperabilidade entre diferentes implementações, garantindo que todas sigam o mesmo conjunto de regras e funcionalidades.

Exemplo: A implementação mais popular do ECMAScript é o JavaScript, usado em navegadores e servidores.

História do ECMAScript

- Criado em 1997 como um padrão para regular o comportamento da linguagem JavaScript.
- ECMAScript 1 foi a primeira versão oficial, padronizando as funcionalidades básicas da linguagem.
- Ao longo dos anos, o padrão evoluiu para incorporar novas funcionalidades e melhorar a linguagem.
- As atualizações começaram a ser anuais a partir de 2015, com o ECMAScript 6 (ES6), também conhecido como ECMAScript 2015.

Importância: O ECMAScript permite que diferentes motores de JavaScript sigam o mesmo comportamento.

Principais versões do ECMAScript

- **ES5 (2009)**:
  - Introduziu o "modo estrito", JSON nativo, e melhorias na manipulação de objetos e arrays.
- **ES6 (2015)**:
  - Considerada uma das maiores atualizações, trouxe classes, módulos, ‘let‘ e ‘const‘, arrow functions, e promises.
- **ES7 (2016)**:
  - Atualização menor, com adição de ‘Array.prototype.includes‘ e o operador de exponenciação (‘**‘).
- **ES8 (2017)**:
  - Introduziu ‘async/await‘ para melhorar o suporte à programação assíncrona.

Versões Recentes do ECMAScript

- **ES9 (2018)**:
  - Incluiu melhorias na manipulação de objetos e assinaturas de funções.
- **ES10 (2019)**:
  - Adicionou ‘Array.prototype.flat‘ e ‘flatMap‘, além de ajustes para strings e revisões no uso de objetos.
- **ES11 (2020)**:
  - Adicionou o operador de encadeamento opcional (‘?.‘), coalescência nula (‘?¿), e importações dinâmicas.
- **Atualizações Anuais**:
  - Pequenos incrementos são feitos todos os anos para manter a linguagem moderna e eficiente.

Nota: A linguagem continua evoluindo com novos recursos planejados a cada ano.

Tipagem no JavaScript

- **Tipagem dinâmica**: Tipo definido em tempo de execução.
- **Tipagem fraca**: Conversão automática de tipos.
- Exemplos:
  - `5 + "5"` $\rightarrow$ `"55"` (concatenação).
  - `5 - "2"` $\rightarrow$ `3` (subtração).
- **Atenção**: Pode causar comportamentos inesperados.

Dica: Use TypeScript para tipagem estática!

Por que 5 + "5" = "55" e 5 - "2" = 3?

- **Concatenação** (`5 + "5"`):
  - O operador + faz a **concatenação** de strings.
  - O número 5 é convertido para string.
  - Resultado: `"55"` (junta os valores como strings).
- **Subtração** (`5 - "2"`):
  - O operador - só funciona com números.
  - A string `"2"` é convertida para o número 2.
  - Resultado: `5 - 2` = 3.

Resumo: JavaScript converte tipos automaticamente para operar com strings ou números.

"Compilação" no JavaScript

- **Interpretada**: Executada linha por linha.
- **JIT Compilation**: Compilação Just-In-Time para otimização.
- Motores como V8 (Chrome) convertem código em nativo durante a execução.

Resultado: Flexibilidade + performance!

TypeScript: Não será utilizado

- **TypeScript**: Superset de JavaScript que adiciona tipagem estática e recursos avançados.
- Facilita a detecção de erros durante o desenvolvimento.
- Oferece suporte para tipos explícitos e funcionalidades como interfaces, classes e enums.
- Amplamente usado em projetos grandes para melhorar a manutenção e previsibilidade do código.

Importante: Apesar dos benefícios, vamos focar em **JavaScript**.

Por que usar JavaScript com HTML?

- **Interatividade**: Responder às ações dos usuários (cliques, digitação, etc.).
- **Dinamicidade**: Atualizar o conteúdo da página sem recarregá-la.
- **Melhor Experiência do Usuário**: Tornar a navegação mais fluida e agradável.
- **Validação de Formulários**: Verificar dados antes de enviar ao servidor.

Inclusão de JavaScript

- **Arquivos Externos**: Melhor prática para organização e manutenção.
- Evita misturar código JavaScript com HTML.

Estrutura do Projeto:

- `index.html`
- `scripts/`
  - `main.js`
  - `main-jquery.js`

Baixar esqueleto projeto:
Projeto de Manipulação de Texto e Cor com jQuery

Plugins - VsCode

- **Plugins que eu uso**
  - ESLint
    - `O que faz: Analisa o código em busca de problemas e más práticas.`
  - Bracket Pair ColorizerPath Intellisense
    - `O que faz: Formata o código automaticamente seguindo regras predefinidas.`
  - Live Server
    - `O que faz: Lança um servidor de desenvolvimento local com recarregamento automático.`

Exemplo 1: Exibir Mensagem ao Clicar em um Botão

Usando Vanilla JavaScript

```
// JavaScript puro
document.getElementById('meuElemento').style.color = 'red';
```

Exemplo 1: Exibir Mensagem ao Clicar em um Botão

Usando jQuery

```
// jQuery
$('#meuElemento').css('color', 'red');
```

Exemplo 2: Alterar o Texto de um Parágrafo

Usando Vanilla JavaScript

```
document.addEventListener('DOMContentLoaded', function() {
    const botao = document.getElementById('alterarBotao');
    const paragrafo = document.getElementById('meuParagrafo');

    botao.addEventListener('click', function() {
        paragrafo.textContent = 'O texto foi alterado!';
    });
});
```

Exemplo 2: Alterar o Texto de um Parágrafo

Usando jQuery

```
$(document).ready(function() {
    $('#alterarBotao').click(function() {
        $('#meuParagrafo').text('O texto foi alterado!');
    });
});
```

Objeto $ e Seletores

- **Objeto $**: Base para manipulação do DOM.
- **Seletores CSS**:
  - Exemplos:
    - $('#id')
    - $('. classe ')
    - $('[atributo]')
- **Boa Prática**: Use seletores específicos para melhor performance.

Manipulação do DOM

- **Métodos Principais**:
  - html(), text(), val(), css()
- **Exemplo**:
  - $('#titulo').text('Novo Ttulo');
- **Boa Prática**: Manipule o DOM após o $(document).ready().

Eventos

- **Eventos Comuns**:
  - click (), hover(), keydown()
- **Exemplo**:
  - $('#botao').on(' click ', function ());
- **Boa Prática**: Prefira on() para compatibilidade.

Animações

- **Efeitos Simples**:
  - fadeIn(), fadeOut(), slideUp()
- **Exemplo**:
  - $('.menu').slideToggle();
- **Boa Prática**: Use animações com moderação.

Requisições AJAX

- **Métodos**:
  - `$.ajax()`, `$.get()`, `$.post()`
- **Exemplo**:
  - `$.get('dados.json', function(data) );`
- **Boa Prática**: Sempre trate erros nas requisições.

Encadeamento e Plugins

- **Encadeamento**:
  - $('#item').addClass('ativo').fadeIn();
- **Plugins**:
  - Extensões como sliders, validação de formulários.
- **Boa Prática**: Use plugins confiáveis e atualizados.

Conclusão

- JavaScript é essencial para criar páginas web interativas.
- Tanto Vanilla JS quanto jQuery têm suas vantagens.
- Pratique os exemplos apresentados para consolidar o aprendizado.

Dica Final: A melhor maneira de aprender é praticando!