Introdução JavaScript

Introdução ao JavaScript

Linguagem de programação client-side e server-side.

Importância no desenvolvimento web.

Diferença entre JavaScript,HTML, e CSS.

História e Evolução
Criação do JavaScript por Brendan Eich em 1995.
Importância do ECMAScript.
Atualizações e versões modernas.

JavaScript

Introdução ao JavaScript

Linguagem de programação client-side e server-side.

Linguagem de programação **client-side** é aquela cujo código é executado diretamente no navegador do usuário (cliente). O código é enviado do servidor para o cliente, onde é processado localmente, permitindo a criação de interfaces interativas e dinâmicas sem a necessidade de comunicação constante com o servidor.

Linguagem de programação **server-side** é aquela cujo código é executado no servidor. O servidor processa as solicitações do cliente, executa o código e retorna a resposta (geralmente em HTML, JSON ou XML) para o navegador do usuário.

•**Client-side:** Código executado no navegador do cliente, responsável pela interatividade e manipulação de elementos visuais da página.
•**Server-side:** Código executado no servidor, responsável por operações como gerenciamento de banco de dados, autenticação, e lógica de negócios antes de enviar os resultados para o cliente.

JavaScript

Importância no desenvolvimento web.

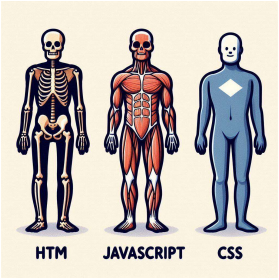
Sistema web é um um software personalizado e hospedado na internet, desenvolvido com base nas necessidades dos usuários, que neste caso são os colaboradores e gestores de uma empresa. O principal objetivo de um sistema web é melhorar a acessibilidade, pois para acessar o software basta entrar com login e senha.

JavaScript

Diferença entre javascript, HTML, e CSS.

Resumo das Diferenças

•**HTML:** Fornece a estrutura da página. Define onde os elementos como cabeçalhos, parágrafos, imagens, e links aparecem.
•**CSS:** Controla o estilo da página. Define a aparência de cada elemento, como cores, tamanhos, espaçamentos e layouts.
•**JavaScript:** Adiciona comportamento à página. Permite interatividade, manipulação dinâmica dos elementos e integração com outros serviços.
Juntos, HTML, CSS e JavaScript criam páginas web completas: HTML define a estrutura, CSS estiliza essa estrutura, e JavaScript traz vida e interatividade à página.



Criação do javascript por Brendan Eich em 1995.

JavaScript é uma das linguagens de programação mais populares no mundo, amplamente utilizada para desenvolvimento web. Sua história remonta aos primeiros dias da internet, quando a necessidade de interatividade nas páginas web estava começando a ser percebida.

Origem e Criação (1995)
•**Netscape e Brendan Eich:**
- Em 1995, a empresa Netscape Communications, uma das pioneiras da internet, estava desenvolvendo um navegador chamado Netscape Navigator. Para tornar as páginas web mais dinâmicas e interativas, a Netscape decidiu criar uma nova linguagem de programação que pudesse ser executada diretamente no navegador.

Ajax e Web 2.0:
•No início dos anos 2000, JavaScript ganhou ainda mais popularidade com o surgimento do conceito de **Ajax** (Asynchronous JavaScript and XML). Ajax permitia que páginas web fossem atualizadas dinamicamente sem recarregar toda a página, proporcionando uma experiência de usuário muito mais fluida. Isso impulsionou a era da **Web 2.0**, marcada por aplicativos web mais interativos e responsivos, como Gmail e Google Maps.

Criação do javascript por Brendan Eich em 1995.

Modernização e Explosão de Popularidade (2010-presente)
•**ECMAScript 5 e 6:**
- A versão **ECMAScript 5** (ES5), lançada em 2009, introduziu várias melhorias significativas, tornando a linguagem mais poderosa e segura. No entanto, foi com **ECMAScript 6** (ES6), também conhecida como **ECMAScript 2015**, que JavaScript realmente se modernizou. ES6 trouxe recursos como classes, módulos, arrow functions, let/const, e muito mais, aproximando JavaScript de outras linguagens de programação modernas.

•**Node.js e JavaScript no Backend:**
- Em 2009, Ryan Dahl lançou **Node.js**, uma plataforma que permite executar JavaScript no lado do servidor, fora do navegador. Isso expandiu enormemente o uso do JavaScript, permitindo que desenvolvedores utilizassem a mesma linguagem tanto no frontend quanto no backend.

Média salarial por Linguagens / Tecnologias

Média salarial por Frameworks / Ferramentas

Mas o que exatamente ele faz na web de hoje?

Imagine um maestro conduzindo uma orquestra, onde cada instrumento tem sua função.
O JavaScript é esse maestro, trazendo harmonia e dinâmica a cada elemento da página.

Porém, para que essa mágica aconteça, ele precisa de uma partitura, um guia. E é aí que entra o DOM (Document Object Model/ Modelo de objeto de documento).

Pense no DOM como a estrutura esquelética de cada página que você visita.

É ele que diz ao JavaScript onde estão os violinos, os cellos, as flautas. O DOM permite que o JavaScript "veja" e interaja com cada elemento da página, tornando possível a manipulação em tempo real.
É como se, a cada vez que você clicasse ou digitasse algo, a orquestra ajustasse sua melodia instantaneamente.

JavaScript

Mas o que exatamente ele faz na web de hoje?

Essa relação simbiótica entre JavaScript e DOM é o que traz vida às páginas web.

Eles dançam juntos, em perfeita sincronia, criando experiências que vão além do visual.

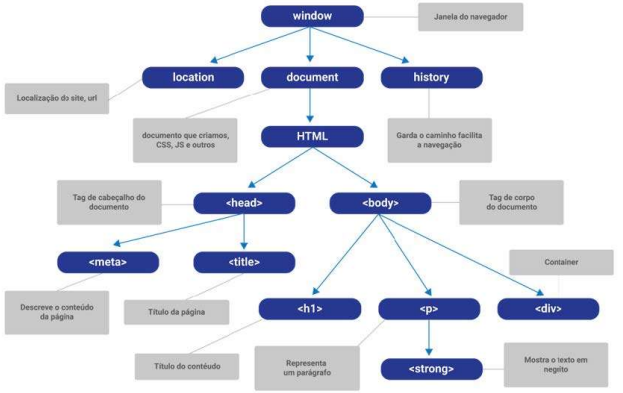
Cada ação e movimento do usuário desencadeia uma resposta.
É essa combinação que torna a navegação tão fluida, tão interativa.

Então, da próxima vez que você estiver navegando e perceber algo incrível acontecendo na página, lembre-se da parceria lendária entre o JavaScript e o DOM.

Eles são, sem dúvida, os artistas por trás de cada grande performance na web!

JavaScript

Árvore Hierárquica DOM



JavaScript

Configuração do Ambiente

•**Editores de Código**
- •Editores: Visual Studio Code.

•**Console do Navegador**
- •Como acessar e utilizar o console do navegador para testar código.

Crie uma pasta em Documentos com seu nome, dentro dela uma pasta com nome Projeto01

Crie um arquivo index.html dentro do body crie o comando para chama o arquivo .js

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

Exemplo prático: console.log('Hello, JavaScript!');

JavaScript

Sintaxe Básica

•**Variáveis**
- •Introdução aos tipos de variáveis: var, let, const.
- •Exemplos de declaração e atribuição de valores:

```
var nome = 'Maria';
let idade = 25;
const PI = 3.14;
```

Tipos de Dados Números, Strings, Booleanos, Arrays, Objetos.
Exemplos:

```
let numero = 10;
let texto = "Olá Mundo";
let verdade = true;
let lista = [1, 2, 3];
let objeto = { nome: "João", idade: 30 };
```

JavaScript

Variáveis em JavaScript representam espaços reservados na memória para armazenar valores.
Na linguagem, existem diferentes maneiras de definir esses espaços: através das palavras-chave var, let ou const.

VAR : A variável declarada com var tem escopo global se declarada fora de uma função, ou escopo de função se declarada dentro de uma função. Ela não tem escopo de bloco (ou seja, não respeita chaves {}).
LET tem escopo de bloco, o que significa que é acessível apenas dentro do bloco onde foi declarada (entre {}).

CONST também tem escopo de bloco.
Hoisting: Como let, const é "hoisted", mas não inicializada, então usá-la antes da declaração causará um ReferenceError.

Outra característica distintiva do JavaScript é sua tipagem dinâmica, o que permite aos desenvolvedores atribuir valores a variáveis sem precisar especificar seu tipo previamente.

JavaScript

Regras e melhores práticas na nomeação

O nome de uma variável pode conter letras, números, underscores (_) e cifrões ($).

O nome deve começar com uma letra, $ ou _.

Não pode começar com um número.

Evite usar nomes reservados da linguagem.

Use camelCase para multiplas palavras.

JavaScript

Usando Condicionais e Loops no JavaScript

O que é condição em JavaScript? Uma condição em JavaScript é uma expressão que pode ser avaliada como verdadeira (true) ou falsa (false). Ela é usada para tomar decisões no código.

Se a condição dentro de um if for avaliada como verdadeira, o código dentro do bloco if será executado.
Quando a condição dentro de um if não é verdadeira, o código dentro do bloco else (se fornecido) é executado.

Podemos usar else if para verificar múltiplas condições em sequência.
O switch é útil quando temos que verificar uma variável contra vários valores.
O break é crucial em um bloco switch para garantir que, após a execução do código para um caso específico, o controle saia do bloco switch. Sem o break, todos os casos subsequentes também serão executados, mesmo se não corresponderem à condição.
O caso default é executado quando nenhum dos outros casos corresponde à condição.

O que são Loops? Loops, ou laços de repetição, são estruturas em programação que permitem que um conjunto de instruções seja repetido várias vezes, baseado em uma condição ou até que uma condição seja atendida. Isso pode ser útil para iterar através de listas, arrays, ou simplesmente para repetir uma operação até que uma condição desejada seja alcançada.

Estrutura básica dos Loops: A maioria dos loops segue um padrão: variável, condição, incremento/decremento, sendo as 3 for, while e do while.

Quando e por que usar?

For: É apropriado quando o número de iterações é conhecido de antemão. Por exemplo, quando se percorre arrays ou listas com um número fixo de elementos.

While: É ideal quando o número de iterações não é conhecido previamente, e o loop precisa ser executado enquanto uma certa condição for verdadeira.

Do While: Útil quando o bloco de código precisa ser executado pelo menos uma vez, independentemente da condição inicial. Em outras palavras, primeiro você faz, depois verifica.

Integrando JavaScript com HTML: Páginas Vivas e Interativas

SENAI
Serviço Nacional de Aprendizagem Industrial
PELO FUTURO DO TRABALHO

Exercícios Práticos

Exercício 1:

Crie um script que peça ao usuário seu nome e exiba uma mensagem de boas-vindas.

Resolução:

```
javascript
Copiar código
// Solicita o nome do usuário
let nomeUsuario = prompt("Qual é o seu nome?");

// Exibe uma mensagem de boas-vindas
alert("Bem-vindo, " + nomeUsuario + "!");
```

JavaScript

SENAI
Serviço Nacional de Aprendizagem Industrial
PELO FUTURO DO TRABALHO

Exercícios Práticos

Exercício 2:

Implemente uma função que receba dois números e retorne o maior deles.

Resolução:

```
javascript
Copiar código
// Função para retornar o maior número entre dois valores
function maiorNumero(num1, num2) {
    if (num1 > num2) {
        return num1;
    } else {
        return num2;
    }
}

// Exemplo de uso:
let numero1 = 5;
let numero2 = 10;
console.log("O maior número é: " + maiorNumero(numero1, numero2));
```

JavaScript