MANUAL DE PROGRAMAÇÃO

Senac Vila Prudente

JavaScript

O que é JavaScript?

JavaScript é uma linguagem de programação utilizada para criar interatividade em páginas web. Sendo interpretada e orientada a objetos, ela permite aos desenvolvedores criarem experiências dinâmicas e responsivas para os usuários.

Inserindo JavaScript em uma Página

Você pode incluir JavaScript em uma página HTML usando a tag <script>.

Para incluir JavaScript em uma página HTML, você utiliza a tag <script>. Isso possibilita a execução de código Javascript diretamente na página, interagindo com o conteúdo HTML e tornando a experiência do usuário mais dinâmica.

```
<script>
  // Seu código JavaScript aqui
</script>
```

Conceitos Básicos

Variáveis e Tipos de Dados

Declaração de variáveis e tipos de dados em JavaScript.

JavaScript utiliza variáveis para armazenar dados. Tipos de dados incluem strings (texto), números, booleanos e outros. A

definição de variáveis é flexível, podendo ser alteradas conforme necessário.

```
let nome = 'John';
const PI = 3.14;
let idade = 25;
let isEstudante = true;
```

Operadores

Operadores aritméticos, de atribuição, de comparação e lógicos.

Operadores em JavaScript são símbolos que executam operações em variáveis e valores. Eles incluem operadores aritméticos para cálculos, operadores de comparação e lógicos para controle de fluxo.

```
let soma = 5 + 3;
let igualdade = idade === 25;
let andLogico = (idade > 18) && (isEstudante === true);
```

Estruturas de Controle

Condicionais (if, else, switch)

Condicionais permitem que o código tome decisões com base em expressões condicionais. Se uma condição for verdadeira, um bloco de código é executado; caso contrário, outro bloco pode ser executado.

Estruturas de controle para tomada de decisões.

```
if (idade >= 18) {
  console.log('É um adulto');
} else {
  console.log('É menor de idade');
}
```

Loops (for, while)

Loops são utilizados para repetir a execução de um bloco de código várias vezes. O for é útil quando o número de iterações é conhecido, enquanto o while é usado quando a condição de parada pode variar.

Estruturas de controle para repetição de código.

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}

let contador = 0;
while (contador < 5) {
  console.log(contador);
  contador++;
}
```

Funções

Declaração de Funções

Funções permitem encapsular blocos de código para reutilização. Elas recebem argumentos, executam instruções e podem retornar valores.

Como criar e chamar funções em JavaScript.

```
function saudacao(nome) {
  return 'Olá, ' + nome + '!';
}

let mensagem = saudacao('João');
console.log(mensagem);
```

Funções Anônimas e Arrow Functions

Além das funções convencionais, JavaScript oferece funções anônimas e arrow functions, formas mais concisas de declarar funções.

Outras formas de declarar funções.

```
let quadrado = function(x) {
  return x * x;
};

let cubo = x => x * x * x;
```

Objetos e Arrays

Objetos

Objetos em JavaScript permitem agrupar dados e comportamentos relacionados. Eles consistem em propriedades (chave-valor).

Como criar e manipular objetos em JavaScript.

```
let pessoa = {
  nome: 'Alice',
  idade: 30,
  isEstudante: false
};

console.log(pessoa.nome);
```

Arrays

Arrays são estruturas de dados que armazenam coleções ordenadas de elementos. Podem ser utilizados para armazenar e manipular conjuntos de dados.

Trabalhando com arrays em JavaScript.

```
let frutas = ['maçã', 'banana', 'laranja'];
frutas.push('morango');
console.log(frutas[2]);
```

Manipulação de DOM

Seleção de Elementos

O DOM (Document Object Model) representa a estrutura da página web. JavaScript é usado para selecionar e interagir com elementos HTML no DOM, possibilitando alterações dinâmicas.

Como selecionar elementos HTML usando JavaScript.

```
let elemento = document.getElementById('meuElemento');
```

Manipulação de Conteúdo

JavaScript pode alterar o conteúdo de elementos HTML, permitindo a atualização dinâmica da página sem a necessidade de recarregamento.

Alterando o conteúdo de elementos HTML.

```
elemento.innerHTML = 'Novo conteúdo';
```

AJAX e Fetch API

Requisições Assíncronas

A Fetch API permite realizar requisições assíncronas para recuperar dados de servidores, sem bloquear a execução do restante do código.

Realizando requisições assíncronas com a Fetch API.

```
fetch('https://api.exemplo.com/dados')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Erro:', error));
```

Tratamento de Erros

Try...Catch

JavaScript oferece estruturas para lidar com erros de forma controlada. O try...catch permite que o código trate exceções, evitando falhas críticas.

Como lidar com erros em JavaScript.

```
try {
  // Código que pode gerar um erro
} catch (error) {
  console.error('Ocorreu um erro:', error);
}
```

Programação Assíncrona e Promises:

JavaScript é assíncrono por natureza. Promises são uma abordagem moderna para lidar com operações assíncronas, permitindo um código mais limpo e legível.

```
function fetchData() {
  return new Promise((resolve, reject) => {
    // Lógica assíncrona aqui
    if (sucesso) {
      resolve(dados);
    } else {
      reject(erro);
    }
  });
}

fetchData()
  .then(result => console.log(result))
  .catch(error => console.error(error));
```

ES6+ (ECMAScript 2015 em diante):

Recursos modernos do JavaScript, como Arrow Functions, Destructuring, Template Literals, e Classes, melhoram a legibilidade e a eficiência do código.

```
// Arrow Function
const square = x => x * x;

// Destructuring
const { nome, idade } = pessoa;

// Template Literals
const mensagem = `Olá, ${nome}! Você tem ${idade} anos.`;

// Classes
class Pessoa {
  constructor(nome, idade) {
    this.nome = nome;
    this.idade = idade;
  }
}
```

Módulos e Import/Export:

JavaScript suporta módulos para organizar o código de maneira modular e reutilizável.

```
// No arquivo modulo.js
export const soma = (a, b) => a + b;

// No arquivo principal
import { soma } from './modulo';
console.log(soma(3, 4));
```

Ferramentas e Bibliotecas Populares:

Node.js:

Ambiente de execução JavaScript do lado do servidor que permite construir aplicativos escaláveis e de alto desempenho.

npm (Node Package Manager):

Gerenciador de pacotes para JavaScript. Permite instalar, compartilhar e gerenciar dependências em projetos.

Webpack:

Ferramenta para empacotar e modularizar recursos da web, facilitando o desenvolvimento e a otimização do código.

React, Angular, Vue:

Bibliotecas/frameworks populares para construir interfaces de usuário reativas e componentizadas.

Express.js:

Framework para construir aplicativos web e APIs usando Node.js.

Práticas Recomendadas:

ESLint e Prettier:

Ferramentas para manter a consistência e a qualidade do código, aplicando padrões de estilo e identação.

Testes (Jest, Mocha, Jasmine):

Desenvolver e manter testes é essencial para garantir a confiabilidade e a robustez do código.

Documentação (JSDoc):

Escrever documentação clara e abrangente facilita a compreensão do código por outros desenvolvedores.

Tendências e Novidades:

WebAssembly (Wasm):

Tecnologia que permite a execução de código de baixo nível na web, abrindo portas para desempenho ainda melhor.

Serverless Architecture:

Arquitetura sem servidor, onde a execução de código é gerenciada automaticamente por provedores de nuvem, como AWS Lambda ou Azure Functions.

TypeScript:

Superset tipado de JavaScript que adiciona tipos estáticos à linguagem, melhorando a segurança e a manutenção do código.

Async/Await:

A sintaxe async/await simplifica a gestão de operações assíncronas, tornando o código mais legível e semelhante à programação síncrona.

```
async function fetchData() {
  try {
    let response = await fetch('https://api.example.com/data');
    let data = await response.json();
    console.log(data);
  } catch (error) {
    console.error('Erro:', error);
  }
}
```

Generators:

Generators permitem pausar e retomar a execução de uma função, facilitando o controle assíncrono.

```
function* gerador() {
  yield 1;
  yield 2;
  yield 3;
}

let gen = gerador();
console.log(gen.next().value); // 1
```

Map, Set, WeakMap, WeakSet:

Estruturas de dados avançadas para manipulação de coleções e conjuntos.

```
let mapa = new Map();
mapa.set('chave', 'valor');

let conjunto = new Set();
conjunto.add('item');
```

Proxy:

Proxy permite interceptar operações em objetos, proporcionando controle mais granular sobre o comportamento de objetos.

```
let handler = {
  get: function(target, prop, receiver) {
    console.log(`Obtendo propriedade: ${prop}`);
    return Reflect.get(target, prop, receiver);
  }
};

let objeto = new Proxy({}, handler);
objeto.nome; // Exibe: Obtendo propriedade: nome
```

Frameworks e Bibliotecas:

Redux (para React):

Biblioteca para gerenciamento de estado em aplicações React, promovendo um estado previsível e facilitando o teste.

Angular CLI (para Angular):

Interface de linha de comando para Angular que facilita a criação, desenvolvimento e teste de projetos Angular.

Vue.js:

Um framework progressivo para a construção de interfaces de usuário. Pode ser adotado gradualmente em projetos existentes.

Ferramentas e Práticas Recomendadas:

Babel:

Transcompilador que permite escrever código JavaScript usando a sintaxe mais recente, garantindo a compatibilidade com versões mais antigas de navegadores.

Webpack e Parcel:

Ferramentas para empacotamento de módulos, otimização de recursos e criação de bundles para produção.

Git e GitHub:

Versionamento de código é uma prática essencial. O Git, junto com plataformas como GitHub, simplifica o controle de versão e colaboração em equipe.

Aprofundamento em Conceitos:

Hoisting:

Compreender como o hoisting afeta a execução do código, onde declarações de variáveis e funções são elevadas ao topo de seus escopos.

Closures:

Entender closures, funções que têm acesso a variáveis fora de seu próprio escopo, é crucial para construir código JavaScript robusto.

Event Loop:

Compreender o modelo de concorrência de JavaScript, que inclui o conceito de pilha de chamadas, callback queue e loop de eventos.

Tendências e Futuro:

Web Components:

Web Components são uma especificação do W3C para criar componentes reutilizáveis personalizados, promovendo a reutilização de código e interoperabilidade.

GraphQL:

Uma alternativa eficiente às APIs REST, GraphQL permite que clientes requisitem dados específicos, reduzindo a sobrecarga de dados transmitidos pela rede.

Machine Learning em JavaScript:

O uso de bibliotecas como TensorFlow.js permite a implementação de modelos de machine learning diretamente no navegador.