UNISAGRADO
Ensino Superior de Excelência

Curso de Introdução ao JS JavaScript

Módulo 1: Introdução ao JavaScript
- História e importância do JS;
- Instalação do VS Code;
- Introdução ao console do navegador.

Módulo 2: Declaração de Variáveis e Operações básicas
- Var, let e const, boas práticas;
- Operações aritméticas;
- Operadores de comparação, concatenação.

Módulo 3: Manipulação do DOM
- O que é o DOM;
- Métodos de busca de elementos;
- Alteração de conteúdo;
- Manipulação de eventos.

Módulo 4: Funções
- Funções tradicionais e arrow functions;
- Parâmetros e retorno.

Módulo 5: Manipulação de Objetos
- Criando e manipulando objetos;
- Arrays de objetos;
- Extraindo valores de objetos.

Módulo 6: Funções assíncronas
- Introdução a assincronia;
- Trabalhando com API's;
- Try, Fetch, Catch.

Módulo 1: Introdução ao JavaScript

História e Importância do JavaScript

JavaScript é uma das linguagens de programação mais populares e amplamente usadas no mundo. Originalmente criada para tornar páginas web mais dinâmicas, ela evoluiu para ser uma linguagem full-stack, possibilitando o desenvolvimento tanto de front-end (interações com o usuário) quanto de back-end (servidores e bancos de dados). Por exemplo, o Node.js, um ambiente de execução de JavaScript do lado do servidor, permite que os desenvolvedores utilizem a mesma linguagem em toda a stack de desenvolvimento.

O primeiro passo no aprendizado de JavaScript é configurar seu ambiente de desenvolvimento. Embora seja possível escrever e executar código diretamente no navegador através da aba "Console" nas ferramentas de desenvolvedor, editores de texto como o VS Code oferecem uma experiência de desenvolvimento mais rica, com suporte a extensões para melhorar a produtividade. Algumas extensões recomendadas incluem ESLint, que ajuda a identificar e corrigir erros no código, e Prettier, que garante a consistência na formatação do código. Esses detalhes, embora pequenos, fazem uma grande diferença no dia a dia de um desenvolvedor profissional.

Entender a sintaxe básica do JavaScript é fundamental antes de avançar para tópicos mais complexos. A linguagem é case-sensitive, ou seja, distingue letras maiúsculas de minúsculas, e utiliza as convenções camelCase para nomear variáveis e funções. Além disso, o JavaScript oferece suporte a uma variedade de tipos de dados primitivos, incluindo números, strings, booleanos, null, undefined, e a partir do ECMAScript 2015 (ES6), o tipo symbol. Todos esses tipos são frequentemente utilizados para representar informações em aplicações reais.

Exemplo Prático:

Imagine que você está desenvolvendo uma aplicação que exibe mensagens personalizadas aos usuários em função da hora do dia. Com JavaScript, você poderia usar o seguinte código simples:

```
let agora = new Date();
let hora = agora.getHours();

if (hora < 12) {
  console.log('Bom dia!');
} else if (hora < 18) {
  console.log('Boa tarde!');
} else {
  console.log('Boa noite!');
}
```

Esse exemplo simples ilustra o uso de variáveis, operadores de comparação e condicionais, que serão usados em praticamente todos os tipos de aplicações.

(O Date() é um construtor que cria um objeto de data e hora atual no JavaScript. Quando usamos new Date(), ele captura o momento exato da criação (data e hora atuais). O método getHours() retorna apenas a hora (0 a 23) do objeto Date.)

Desafio

Crie um script simples que simule um relógio digital. O script deve exibir a hora atual no formato HH:MM:SS e atualizar a cada segundo.

- Utilize o objeto Date para obter a hora atual.
- Use setInterval para fazer o relógio atualizar automaticamente.
- Use condicionais para garantir que os minutos e segundos sempre tenham dois dígitos (ex: 09 em vez de 9).

Exemplo de Saída:

```
10:08:45
```

Módulo 2: Declaração de Variáveis e Operações Básicas

No JavaScript, as variáveis são recipientes que armazenam dados que podem ser usados posteriormente no programa. Existem três formas principais de declarar variáveis: var, let, e const.

- **var**: O escopo de **var** é a função ou o contexto global, o que pode gerar comportamentos inesperados devido ao hoisting (quando a declaração de variáveis é movida para o topo do escopo durante a fase de compilação). Por isso, é menos utilizado em códigos modernos.
- **let**: Introduzido no ECMAScript 6 (ES6), **let** tem escopo de bloco e é geralmente preferível ao var, pois oferece mais previsibilidade no comportamento da variável.
- **const**: Também introduzido no ES6, **const** é usado para declarar variáveis cujo valor não será alterado. Ele garante que o valor atribuído à variável permaneça constante, o que é útil quando você quer garantir a imutabilidade de um dado.

Além da declaração de variáveis, o JavaScript possui um conjunto completo de operadores aritméticos e lógicos. Os operadores aritméticos incluem o básico como adição (**+**), subtração (**-**), multiplicação (*****), divisão (**/**), e o operador de módulo (**%**), que retorna o resto de uma divisão. Já os operadores lógicos, como **&&** (E lógico), **||** (OU lógico), e **!** (negação), são amplamente utilizados em expressões condicionais, facilitando a tomada de decisões no código.

Além disso, o JavaScript também possui operadores de comparação, que são frequentemente usados para verificar se valores são iguais (==, ===), diferentes (!=, !==), maiores ou menores (>, <), ou equivalentes. A diferença entre == e === é fundamental: == compara apenas o valor, enquanto === compara tanto o valor quanto o tipo de dado.

Exemplo Prático:

Um caso muito comum no mercado é o desenvolvimento de sistemas que calculam o preço total de uma compra, incluindo descontos e impostos. Suponha que você tenha o seguinte cenário:

```
let precoProduto = 200;
let desconto = 0.1; // 10% de desconto
let precoComDesconto = precoProduto - (precoProduto * desconto);
let imposto = 0.15; // 15% de imposto
let precoFinal = precoComDesconto + (precoComDesconto * imposto);

console.log(`O preço final do produto é R$ ${precoFinal}`);
```

Esse código exemplifica o uso de variáveis, operadores aritméticos e a importância de organizar cálculos corretamente em um sistema que será usado no mundo real, como um sistema de e-commerce.

Desafio

Crie um programa que receba três números como entrada, calcule a média deles e determine se a média é maior, menor ou igual a 50. Exiba uma mensagem correspondente para cada caso.

- Utilize prompt() para capturar entradas do usuário (ou simule no console).
- Converta os valores de string para número usando Number() ou parseInt().
- Use operadores de comparação para verificar o valor da média.

Exemplo de Saída:

```
A média é 65, portanto é maior que 50.
```

Módulo 3: Manipulação Básica do DOM

O **DOM** (Document Object Model) é uma interface que representa documentos HTML ou XML em forma de árvore de nós, onde cada nó é uma parte da página: elementos, atributos, textos, etc. Através do DOM, podemos modificar o conteúdo e a estrutura de uma página dinamicamente, o que torna o JavaScript tão poderoso para a construção de interfaces interativas.

A manipulação do DOM é feita, principalmente, através de métodos de seleção de elementos e de eventos. Métodos como **getElementById**, **querySelector**, e **querySelectorAll** permitem acessar qualquer elemento de uma página. Uma vez que os elementos são selecionados, podemos alterar seu conteúdo ou estilo diretamente usando propriedades como **innerHTML**, **textContent**, e **style**.

Outro aspecto importante da manipulação do DOM são os eventos. Eventos permitem que os desenvolvedores configurem interações que ocorrem quando o usuário realiza ações específicas, como clicar em um botão ou passar o mouse sobre um elemento. Para isso, utilizamos o método **addEventListener**, que permite "ouvir" eventos e reagir a eles através de funções callback.

"O DOM foi criado pela W3C com o objetivo de desenvolver um padrão para linguagens de script para os navegadores já que antigamente cada navegador tinha seu próprio modo de manipular os objetos, o que gerava muita incompatibilidade e obrigava os desenvolvedores a escrever uma versão de script para cada navegador."

O que é DOM? A árvore de elementos do HTML.
https://tableless.github.io/iniciantes/manual/obasico/oquedom.html#:~:text=O%20DOM%20foi%20criado%20pela,de%20script%20para%20cada%20navegador.

Exemplo Prático:

Um exemplo clássico de manipulação do DOM é a implementação de um sistema de formulário que valida entradas de usuário em tempo real. Vamos supor que você tenha um formulário de login onde o usuário deve inserir seu nome e senha. Com JavaScript, você pode verificar se o campo de senha foi preenchido corretamente:

```html
<form id="loginForm">
    <input type="text" id="username" placeholder="Nome de usuário">
    <input type="password" id="password" placeholder="Senha">
    <button type="submit">Entrar</button>
  </form>

  <script>
    document.getElementById('loginForm').addEventListener('submit', function(event) {
      let senha = document.getElementById('password').value;
      if (senha.length < 6) {
        alert('A senha deve ter pelo menos 6 caracteres.');
        event.preventDefault(); // Impede o envio do formulário
      }
    });
  </script>
```

Esse código exemplifica a validação de formulários, algo amplamente utilizado em qualquer aplicação web que interaja com dados de usuários. Ao interceptar o evento de envio do formulário, você pode verificar as entradas e impedir ações indesejadas.

Desafio

Crie um programa que receba três números como entrada, calcule a média deles e determine se a média é maior, menor ou igual a 50. Exiba uma mensagem correspondente para cada caso.

- Use addEventListener para adicionar ouvintes de eventos aos botões.
- Modifique o estilo da página alterando a propriedade backgroundColor do elemento body.

Exemplo: Ao clicar no botão **Verde**, a cor de fundo da página deve mudar para verde.

Módulo 4: Funções em JavaScript

As funções em JavaScript são blocos de código que podem ser executados sempre que forem invocados. Elas são fundamentais para a reutilização e organização do código, permitindo encapsular lógicas que podem ser chamadas várias vezes em diferentes partes de uma aplicação.

Há diferentes maneiras de declarar funções em JavaScript. A maneira tradicional usa a palavra-chave **function**, mas o ECMAScript 6 introduziu as **arrow functions**, que têm uma sintaxe mais curta e um comportamento diferente em relação ao escopo de **this**. Esse comportamento é particularmente útil quando trabalhamos com eventos ou funções de callback, pois evita o comportamento inesperado de **this** comum nas funções tradicionais.

Funções podem receber parâmetros e retornar valores, permitindo que dados sejam passados entre diferentes partes do programa. Além disso, em JavaScript, funções são **first-class citizens** (ou de primeira classe), o que significa que podem ser tratadas como valores, passadas como argumentos para outras funções, retornadas de funções e atribuídas a variáveis.

Basicamente, “Funções são blocos de construção fundamentais em JavaScript. Uma função é um procedimento de JavaScript - um conjunto de instruções que executa uma tarefa ou calcula um valor. Para usar uma função, você deve defini-la em algum lugar no escopo do qual você quiser chamá-la.”

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Functions

Exemplo Prático:

Em um sistema de gestão de pedidos de uma loja online, funções são amplamente utilizadas. Imagine que você precise calcular o valor de um pedido com base no subtotal e nas taxas aplicáveis:

```
function calcularTotal(subtotal, taxa) {
    return subtotal + (subtotal * taxa);
  }

  let pedido1 = calcularTotal(100, 0.1); // Pedido de 100 reais com 10% de taxa
  let pedido2 = calcularTotal(200, 0.15); // Pedido de 200 reais com 15% de taxa

  console.log(`Total do pedido 1: R$ ${pedido1}`);
  console.log(`Total do pedido 2: R$ ${pedido2}`);
```

Aqui, a função **calcularTotal** encapsula a lógica do cálculo de um pedido, permitindo reutilizar esse código sempre que necessário, com diferentes parâmetros.

Desafio

Escreva uma função chamada **calcularDesconto** que receba o valor de um produto e a porcentagem de desconto, e retorne o valor final com o desconto aplicado. Use a função para calcular o preço final de três produtos diferentes e exiba os resultados no console.

- Use a função para processar diferentes valores de produtos.
- A função deve ter dois parâmetros: o valor original e a porcentagem de desconto.

Exemplo de saída:

```
O valor final do produto 1 é R$90.
O valor final do produto 2 é R$180.
O valor final do produto 3 é R$270.
```

Módulo 5: Manipulação de Objetos

Objetos em JavaScript são coleções de dados e funcionalidades, organizados em pares chave-valor. Eles são uma das estruturas de dados mais utilizadas, uma vez que representam bem a realidade em diversos cenários: um objeto pode representar um produto, um usuário, uma transação, etc. Cada chave do objeto é um identificador único, e o valor pode ser de qualquer tipo, inclusive outros objetos ou funções, conhecidos como métodos.

Manipular objetos é uma habilidade crucial no mercado de trabalho, especialmente em projetos que envolvem grandes volumes de dados estruturados. A capacidade de acessar, modificar e iterar sobre objetos, assim como a habilidade de utilizar métodos nativos como **Object.keys()**, **Object.values()**, e **Object.entries()**, são conhecimentos essenciais para lidar com dados complexos.

Resumidamente, “A linguagem JavaScript é projetada com base em um simples paradigma orientado a objeto. Um objeto é uma coleção de propriedades, e uma propriedade é uma associação entre um nome (ou chave) e um valor. Um valor de propriedade pode ser uma função, que é então considerada um método do objeto”
https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Working_with_objects

“Objetos são como uma espécie de "super variáveis" que armazenam uma "coleção de valores" referenciados por nome, e que podem ser recuperados para serem utilizados em diversas outras partes de um programa. Em JavaScript praticamente qualquer tipo de dado é um objeto.”
https://tableless.github.io/iniciantes/manual/js/objetos.html#:~:text=Objetos%20são%20como%20uma%20espécie,de%20dado%20é%20um%20objeto.

Exemplo Prático:

Em uma aplicação de gerenciamento de estoque, cada produto pode ser representado por um objeto com propriedades como nome, preço e quantidade:

Esse exemplo demonstra como objetos podem armazenar tanto dados quanto funcionalidades (métodos) associados a esses dados. Ao construir sistemas complexos, como um software de gestão de estoques, você estará manipulando objetos e métodos constantemente.

```javascript
let produto = {
    nome: "Notebook",
    preco: 2500,
    quantidade: 5,
    vender: function() {
      if (this.quantidade > 0) {
        this.quantidade--;
        console.log(`Venda realizada. Quantidade restante: ${this.quantidade}`);
      } else {
        console.log("Produto esgotado.");
      }
    }
  };

  produto.vender(); // Chamando o método vender
  produto.vender(); // Chamando o método vender de novo
```

Desafio

Crie um objeto que represente um carro. O objeto deve conter as seguintes propriedades: **marca**, **modelo**, **ano**, **velocidadeAtual** e um método **acelerar** que aumente a velocidade em 10 km/h. Adicione também um método **frear** que reduza a velocidade em 10 km/h, mas nunca permitindo que a velocidade fique abaixo de 0.

- Defina o método acelerar para incrementar a velocidadeAtual.
- O método frear deve usar uma condicional para evitar valores negativos.

Exemplo:

```javascript
carro.acelerar(); // Velocidade atual: 10 km/h
carro.acelerar(); // Velocidade atual: 20 km/h
carro.frear(); // Velocidade atual: 10 km/h
```

Módulo 6: Funções Assíncronas

Funções assíncronas são um aspecto crucial do desenvolvimento em JavaScript, especialmente quando se trata de tarefas que envolvem operações de rede, como consultas a APIs, ou operações de leitura de arquivos e bancos de dados. O JavaScript é baseado em um modelo de execução assíncrona, o que significa que ele pode realizar várias tarefas ao mesmo tempo sem bloquear o fluxo do programa. No entanto, isso cria desafios para lidar com a execução de código que depende do resultado de uma operação assíncrona.

Tradicionalmente, JavaScript lidava com operações assíncronas usando callbacks, mas isso frequentemente resultava no famoso callback hell, onde funções aninhadas dificultavam a leitura e manutenção do código. Para resolver esse problema, o ECMAScript 6 introduziu Promises, que permitem uma abordagem mais organizada para o tratamento de operações assíncronas. Posteriormente, o async/await foi introduzido, permitindo escrever código assíncrono de maneira muito mais clara e legível, como se fosse código síncrono.

Basicamente, “Uma função assíncrona consiste em criarmos uma função “normal”, mas que ganhará como que “superpoderes”, tornando-se uma função capaz de trabalhar com código assíncrono. Como isso é possível? Podemos começar analisando o que as palavras async e await representam.”

https://hcode.com.br/blog/javascript-assincrono-o-guia-completo-parte-5-entendendo-async-await#:~:text=Uma%20função%20assíncrona%20consiste%20em,palavras%20async%20e%20await%20representam.

Exemplo Prático:

Imagine que você esteja desenvolvendo um aplicativo de clima que precisa buscar dados de uma API. Com async/await, isso pode ser feito de forma mais legível:

```
1  async function obterClima(cidade) {
2      try {
3        let resposta = await fetch(`https://api.exemplo.com/clima?cidade=${cidade}`);
4        let dados = await resposta.json();
5        console.log(`O clima em ${cidade} é ${dados.temperatura} graus.`);
6      } catch (erro) {
7        console.error("Erro ao obter os dados do clima:", erro);
8      }
9    }
10
11   obterClima("São Paulo");
```

Esse código exemplifica como operações assíncronas, como chamadas a uma API, podem ser gerenciadas de forma eficiente e sem a complexidade dos callbacks. No mercado, esse tipo de funcionalidade é comum em aplicações que precisam interagir com serviços externos, como plataformas de pagamento, APIs de dados ou integrações de terceiros.

Desafio

Utilize a função **fetch()** para consumir dados de uma API pública (por exemplo, uma API de cotação de moedas). Crie uma função assíncrona que faça a requisição e exiba no console a cotação atual de uma moeda (ex: dólar).

- Use async/await para lidar com a resposta da API.
- Converta a resposta da API para JSON usando .json().
- Exiba a cotação da moeda no console.

Exemplo de saída:

```
1  A cotação atual do dólar é: R$ 5.30
```

Solução dos desafios

Desafio 1: Criar um relógio digital

```
function atualizarRelogio() {
    const agora = new Date();
    const horas = String(agora.getHours()).padStart(2, '0');
    const minutos = String(agora.getMinutes()).padStart(2, '0');
    const segundos = String(agora.getSeconds()).padStart(2, '0');
    console.log(`${horas}:${minutos}:${segundos}`);
}

setInterval(atualizarRelogio, 1000); // Atualiza o relogio de 1 em 1 segundo.
atualizarRelogio();

```

Desafio 2: Calcular a média e comparar com 50

```
function calcularMedia(n1, n2, n3) {
    return (n1 + n2 + n3) / 3;
}

function compararCom50(media) {
    if (media > 50) {
        console.log(`A média é ${media.toFixed(2)}, portanto é maior que 50.`);
    } else if (media < 50) {
        console.log(`A média é ${media.toFixed(2)}, portanto é menor que 50.`);
    } else {
        console.log(`A média é ${media.toFixed(2)}, portanto é igual a 50.`);
    }
}

let numero1 = 45;
let numero2 = 55;
let numero3 = 65;
let media = calcularMedia(numero1, numero2, numero3);
compararCom50(media);

```

Solução dos desafios

Desafio 3: Mudar a cor de fundo da página

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Troca de Cores</title>
</head>
<body>
    <button id="btnVerde">Verde</button>
    <button id="btnVermelho">Vermelho</button>
    <button id="btnAzul">Azul</button>

    <script>
        document.getElementById('btnVerde').addEventListener('click', function() {
            document.body.style.backgroundColor = 'green';
        });
        document.getElementById('btnVermelho').addEventListener('click', function() {
            document.body.style.backgroundColor = 'red';
        });
        document.getElementById('btnAzul').addEventListener('click', function() {
            document.body.style.backgroundColor = 'blue';
        });
    </script>
</body>
</html>
```

Solução dos desafios

Desafio 4: Calcular o valor final com desconto

```
function calcularDesconto(valorOriginal, porcentagemDesconto) {
    return valorOriginal - (valorOriginal * porcentagemDesconto);
}

let precoProduto1 = calcularDesconto(100, 0.1);
let precoProduto2 = calcularDesconto(200, 0.15);
let precoProduto3 = calcularDesconto(300, 0.2);

console.log(`O valor final do produto 1 é R$ ${precoProduto1.toFixed(2)}`);
console.log(`O valor final do produto 2 é R$ ${precoProduto2.toFixed(2)}`);
console.log(`O valor final do produto 3 é R$ ${precoProduto3.toFixed(2)}`);

```

Desafio 5: Criar e manipular um objeto de carro

```
let carro = {
    marca: "Toyota",
    modelo: "Corolla",
    ano: 2021,
    velocidadeAtual: 0,
    acelerar: function() {
        this.velocidadeAtual += 10;
        console.log(`Velocidade atual: ${this.velocidadeAtual} km/h`);
    },
    frear: function() {
        this.velocidadeAtual = Math.max(0, this.velocidadeAtual - 10);
        console.log(`Velocidade atual: ${this.velocidadeAtual} km/h`);
    }
};

carro.acelerar();
carro.acelerar();
carro.frear();
carro.frear();
```

Solução dos desafios

Desafio 6: Consumir dados de uma API

```javascript
const API_KEY = 'Aqui você deve inserir sua API KEY';

async function obterCotacao() {
    try {
        let resposta = await fetch(`https://v6.exchangerate-api.com/v6/${API_KEY}/latest/USD`);
        let dados = await resposta.json();
        let cotacao = dados.conversion_rates.BRL;
        console.log(`A cotação atual do dólar é: R$ ${cotacao}`);
    } catch (erro) {
        console.error("Erro ao obter a cotação:", erro);
    }
}

obterCotacao();
```

É importante ressaltar que para este desafio, iremos usar uma chave de uma API, no caso a API ExchangeRate-API.

Para obtermos a chave da API, é necessário apenas informar um e-mail para que esta chave chegue em seu e-mail.

URL da API: https://v6.exchangerate-api.com/v6/${API_KEY}/latest/USD

Cuidado: É importante manter sua chave de API em segurança e não expô-la em código público. Para ambientes de produção, considere usar variáveis de ambiente para gerenciar suas credenciais.

Conclusão

Chegamos ao final do nosso curso de **JavaScript**, e eu gostaria de expressar minha sincera gratidão a todos vocês que acompanharam e participaram ativamente até o final. Foi um prazer compartilhar conhecimentos e explorar juntos os fundamentos e aplicações do **JavaScript**. Espero que os desafios e exemplos tenham sido úteis para consolidar seu aprendizado e que agora vocês se sintam mais confiantes para aplicar o **JavaScript** em projetos reais.
Continuem praticando e explorando, pois a jornada de aprendizado nunca para. Lembre-se de que o domínio de uma linguagem de programação é um processo contínuo e enriquecedor. Estou certo de que todos temos um futuro promissor pela frente na área de desenvolvimento web. Todos nós buscamos aprender cada dia mais!

Desejo a todos muito sucesso e crescimento em suas carreiras e projetos.

Atenciosamente,

Tulio Braga Alves

Material Complementar

Para aprofundar seus conhecimentos, recomendo os seguintes cursos complementares:

- Curso de HTML e CSS do Gustavo Guanabara
- Curso de JavaScript do Gustavo Guanabara
- Curso de Lógica de Programação do Gustavo Guanabara

Esses cursos fornecem uma base sólida e complementar que ajudará a expandir ainda mais suas habilidades.

1 ”Tudo tem o seu tempo determinado, e há tempo para todo o propósito debaixo do céu.”