Resumo - O que foi Aprendido no Módulo 01

JavaScript é uma linguagem Universal:

JavaScript é uma linguagem de programação amplamente utilizada que funciona em todos os principais navegadores e é compatível com sistemas operacionais diferentes. Isso a torna uma escolha universal para o desenvolvimento web.

Com JS, você consegue realizar no Browser:

- **Melhorar a Experiência do Usuário:** JavaScript permite criar interfaces de usuário dinâmicas e responsivas, tornando as páginas da web mais atraentes e funcionais.

- **Adicionar interatividade às páginas:** Com JavaScript, é possível criar elementos interativos como botões, formulários e carrosséis, proporcionando uma experiência mais envolvente para os usuários.

- **Buscar e Exibir Dados:** JavaScript pode ser usado para buscar e exibir dados em tempo real de servidores, atualizando o conteúdo da página sem a necessidade de recarregamento.

Essas são apenas algumas das capacidades do JavaScript, que é uma linguagem essencial para o desenvolvimento web moderno.

O que é um Servidor:

É um computador que armazena e fornece os arquivos necessários para um website funcionar.

Ciclo Request-Response:

Quando um usuário digitar um URL em um navegador, este envia uma **solicitação (request)** para o servidor da web correspondente. O servidor processa a solicitação e envia uma **resposta (response)** que contém os recursos necessários para renderizar a página da web no navegador do usuário.

Esse ciclo é fundamental para a navegação na web.

Adicionando JavaScript a uma Página HTML:

O JavaScript pode ser incluído em uma página HTML dentro das tags ***<script>***.

Ele pode estar localizado tanto no <head> quanto no <body> do documento, mas é recomendável colocar as tags <script> no final da tag <body> para evitar atrasos na renderização da página.

Recomendação para Incluir JavaScript: É uma prática recomendada adicionar tags <script> no final da tag ***<body>*** *para melhorar o desempenho da página, pois permite que o conteúdo HTML seja carregado antes de processar o JavaScript, evitando atrasos visíveis para o usuário.*

<script src= "app.js"></script>

Variáveis (var, const, let):

Variáveis armazenam valores para uso posterior no código.

Const:

Constantes, declaradas com "const", representam valores imutáveis, como const points = 100. Tentar atribuir um novo valor a uma constante resultará em erro.

Let:

Variáveis podem ser declaradas usando "let" seguido do nome da variável e o valor atribuído, como let age = 31, ou podem ser reatribuídas posteriormente, como age = 32.

Comentários no Código:

Comentários são notas no código que não afetam sua execução e são usados para explicar o código.

- Quando o comentário tiver mais de uma linha usar o: /* (ao abrir) *\ (ao fechar)
- Quando o comentário tiver apenas uma linha: //

Tipos de Dados em JavaScript - Number:

Representa valores numéricos, incluindo inteiros e decimais.

Exemplo: let idade = 25;.

Tipos de Dados em JavaScript - Strings:

Armazenam sequências de caracteres, como texto, entre aspas simples ou duplas.

Exemplo: let nome = "João";.

Tipos de Dados em JavaScript - Boolean:

Lida com valores verdadeiro (true) ou falso (false) para lógica.

Exemplo: let isAtivo = true;.

Tipos de Dados em JavaScript - Null:

Indica que uma variável está intencionalmente vazia.

Exemplo: let valor = null;

Tipos de Dados em JavaScript - Undefined:

Variáveis sem valor atribuído são automaticamente undefined.

Exemplo: let variavel;.

Tipos de Dados em JavaScript - Object:

Armazena estruturas de dados complexas.

Exemplo: let pessoa = { nome: "Maria", idade: 30 };.

Tipos de Dados em JavaScript - Symbol:

Usado em associação com objetos para criar propriedades únicas.

Exemplo: const chave = Symbol('chave');.

Tipos de Dados em JavaScript - BigInt:

Usado para representar números inteiros muito grandes.

Exemplo: const bigNumber = 123456789012345678901234567890123456789012345678901234567890n;.

Propriedades e Métodos em STRINGS:

Strings em JavaScript possuem uma variedade de métodos e propriedades para manipulação e consulta de caracteres:

- **length:** Propriedade que retorna o número de caracteres em uma string:

```
let frase = "Olá, Mundo!";
let tamanho = frase.length; // Retorna 12
```

- **charAt(index):** Método que retorna o caractere na posição especificada.

```
let frase = "Olá, Mundo!";
let terceiroCaractere = frase.charAt(2); // Retorna "á"
```

- **substring(start, end):** Método que extrai uma parte da string com base em índices de início e fim.

```
let frase = "Olá, Mundo!";
let parte = frase.substring(0, 3); // Retorna "Olá"
```

- **toUpperCase() e toLowerCase():** Métodos para converter a string para maiúsculas ou minúsculas.

```
let palavra = "Texto";
let maiusculas = palavra.toUpperCase(); // Retorna "TEXTO"
```

- **indexOf(substring):** Método para encontrar a primeira ocorrência de uma substring na string.

```
let frase = "Olá, Mundo!";
let posicao = frase.indexOf("Mundo"); // Retorna 5
```

- **replace(oldStr, newStr):** Método que substitui uma substring específica por outra na string.

```
let frase = "Olá, Mundo!";
let novaFrase = frase.replace("Mundo", "Planeta"); // Retorna "Olá, Planeta!"
```

- **slice(start, end):** Método que extrai uma parte da string com base em índices de início e fim, semelhante ao substring.

```
let frase = "Olá, Mundo!";
let parte = frase.slice(0, 3); // Retorna "Olá"
```

- **split(separator):** Método que divide a string em um array de substrings com base em um separador especificado.

```
let lista = "maçã,banana,uva";
let frutas = lista.split(","); // Retorna ["maçã", "banana", "uva"]
```

Esses métodos são ferramentas úteis para manipular e trabalhar com strings em JavaScript, tornando a linguagem flexível para o processamento de texto.

Arrays:

São estruturas de dados que permitem armazenar e manipular conjuntos de valores. Aqui estão algumas propriedades e operações com exemplos:

- **Declaração de um Array:** Um array pode ser declarado usando colchetes [] e elementos separados por vírgula.

```
let frutas = ["maçã", "banana", "uva"]
```

- **Acesso aos Elementos:** Os elementos do array podem ser acessados usando índices baseados em zero.

```
let primeiraFruta = frutas[0]; // Retorna "maçã"
```

- **Length:** A propriedade length retorna o número de elementos em um array.

```
let numFrutas = frutas.length; // Retorna 3
```

- **Adição e Remoção de Elementos:**
    - **push()** adiciona elementos ao final do array.
    - **pop()** remove o último elemento do array.

```
frutas.push("laranja"); // Adiciona "laranja" ao final
frutas.pop(); // Remove "laranja" do final
```

Métodos de Manipulação em Arrays:

Arrays possuem métodos embutidos para operações como push(), pop(), shift(), unshift(), splice(), entre outros.

```
frutas.shift(); // Remove o primeiro elemento ("maçã")
frutas.unshift("pêra"); // Adiciona "pêra" no início
frutas.splice(1, 0, "morango"); // Insere "morango" na segunda posição
```

- **Busca em Arrays:** O método indexOf() retorna o índice do primeiro elemento correspondente, ou -1 se não encontrado.

```
let indiceBanana = frutas.indexOf("banana"); // Retorna 1
```

- **Concatenação de Arrays:** O método concat() combina dois ou mais arrays em um novo.

```
let outrasFrutas = ["abacaxi", "mamão"];
let todasFrutas = frutas.concat(outrasFrutas);
```

- **Ordenação de Arrays:** O método sort() classifica os elementos de um array (por padrão, em ordem alfabética).

```
frutas.sort(); // Ordena as frutas em ordem alfabética
```

Os arrays são uma parte essencial da programação em JavaScript e são amplamente utilizados para armazenar e manipular conjuntos de dados de forma eficaz.

Aritmética com Number:

Aritmética em JavaScript envolve a realização de operações matemáticas com números. Aqui estão os principais operadores e conceitos relacionados à aritmética em JavaScript:

- Adição (+): Soma dois valores.
- Subtração (-): Subtrai o segundo valor do primeiro.
- Multiplicação (*): Multiplica dois valores.
- Divisão (/): Divide o primeiro valor pelo segundo.
- Módulo (%): Retorna o resto da divisão do primeiro valor pelo segundo.

```
let soma = 5 + 3; // 8
let subtracao = 10 - 4; // 6
let multiplicacao = 3 * 6; // 18
let divisao = 15 / 3; // 5
let resto = 10 % 3; // 1
```

- Ordem das Operações: As operações seguem a ordem padrão da matemática (PEMDAS/BODMAS), onde parênteses têm prioridade, seguidos de exponenciação, multiplicação e divisão, e finalmente adição e subtração.

```
let resultado = 2 * (3 + 4); // 14 (parênteses primeiro)
```

- Incremento e Decremento: Os operadores de incremento (++) e decremento (--) adicionam 1 ou subtraem 1 de uma variável.

```
let numero = 5;
numero++; // Incrementa para 6
numero--; // Decrementa para 5 novamente
```

- Operações com Variáveis: Variáveis podem ser usadas em operações aritméticas.

```
let a = 10;
let b = 5;
let soma = a + b; // 15
```

- Conversão de Tipos: JavaScript realiza conversões automáticas de tipos em operações com diferentes tipos de dados.

```
let texto = "10";
let numero = 5;
let resultado = texto + numero; // "105" (concatenação)
```

- NaN (Not-a-Number): Operações inválidas resultam em NaN.

```
let resultado = "Texto" / 2; // NaN
```

A aritmética em JavaScript é fundamental para realizar cálculos e operações matemáticas em programas e páginas da web interativas.

É importante entender a ordem das operações e como os operadores funcionam para criar lógica eficaz em seu código JavaScript.

Converter “String” em Número:

Converter uma string em um número em JavaScript pode ser feito usando várias técnicas:

- **parseInt():** Este método converte uma string em um número inteiro. Ele analisa a string até encontrar um caractere que não seja um dígito numérico e retorna o valor convertido.

```
let numeroTexto = "123";
let numero = parseInt(numeroTexto); // Converte para 123
```

- **parseFloat():** Semelhante ao parseInt(), mas converte uma string em um número decimal (com ponto flutuante).

```
let numeroTexto = "3.14";
let numero = parseFloat(numeroTexto); // Converte para 3.14
```

- **Number():** A função Number() converte uma string em um número, seja inteiro ou decimal.

```
let numeroTexto = "42";
let numero = Number(numeroTexto); // Converte para 42
```