



PROGRAMAÇÃO WEB FRONT-END



Programação Web Front-End

Aula 1 - JavaScript

Profa. Rosangela de Fátima Pereira Marquesone
romarquesone@utfpr.edu.br

Proposta: apresentar as características, o histórico e as funcionalidades da linguagem de programação JavaScript.

Objetivos: espera-se que após essa aula, você tenha habilidade para compreender os seguintes tópicos:

1. [Descobrir a história da linguagem JavaScript](#)
2. [Compreender as formas de uso da linguagem JavaScript](#)
3. [Conhecer as características da linguagem JavaScript](#)

Dicas de aprendizado:

- Execute todos os passos com atenção, compreendendo o que está sendo realizado;
- Procure não copiar código, para ter a prática de digitar o código desenvolvido;
- Pergunte quando tiver alguma dúvida;
- Mantenha um histórico dos códigos desenvolvidos, seja no github ou em algum outro meio de armazenamento (e-mails, google drive, etc.);
- Tenha curiosidade e explore os recursos apresentados.

Tópicos anteriores:

- Compreender o que é HTML
- Compreender o que são tags HTML básicas
- Criar um arquivo .html no Visual Studio (VS) Code
- Abrir o arquivo .html em um navegador
- Visualizar o código-fonte de uma página em um navegador
- Inspecionar a página em um navegador
- Utilizar o Live Server no VS Code
- Aprender a utilizar tags semânticas
- Aprender a inserir links
- Aprender a inserir listas
- Aprender a criar uma página com seu Curriculum Vitae (CV) (atividade prática)
- Aprender a inserir figuras
- Aprender a utilizar a tag semântica <figure>
- Inserir figuras em seu Curriculum Vitae (CV) (atividade prática)
- Aprender a criar formulários
- Criar um formulário (atividade prática)
- Descobrir o que é CSS
- Aprender a sintaxe do CSS

- Aprender os tipos de seletores CSS
- Aprender as formas de inclusão de CSS
- Aprender a definir cores
- Aprender a alterar as propriedades de texto
- Aprender o conceito de modelo de caixa do CSS
- Aprender a trabalhar com a margem
- Aprender a trabalhar com a borda
- Aprender a trabalhar com o preenchimento (padding)
- Aprender a usar a propriedade display
- Aprender a utilizar a propriedade float
- Aprender a utilizar a propriedade overflow
- Estruturar páginas por meio do modelo de caixa (atividade prática)
- Aprender o conceito de flex-box
- Aprender as propriedades do elemento pai (flex container)
- Aprender as propriedades dos elementos filhos (flex items)

Passo 1 - Descobrir a história da linguagem JavaScript

Iniciaremos esse tutorial identificando o que é JavaScript, e como essa linguagem foi criada.

Considerada uma linguagem de programação de alto nível, JavaScript é utilizada para criar páginas Web dinâmicas e interativas.

Segundo o [índice TIOBE](#), ela é considerada a sexta linguagem mais utilizada no mundo, sendo que em 2014 ela ocupava a décima segunda posição.

Figura 1 - Ranking das linguagens de programação - Novembro de 2024

Nov 2024	Nov 2023	Change	Programming Language	Ratings	Change
1	1		 Python	22.85%	+8.69%
2	3		 C++	10.64%	+0.29%
3	4		 Java	9.60%	+1.26%
4	2		 C	9.01%	-2.76%
5	5		 C#	4.98%	-2.67%
6	6		 JavaScript	3.71%	+0.50%
7	13		 Go	2.35%	+1.16%

Fonte: <https://www.tiobe.com/tiobe-index/>

Embora JavaScript esteja em alta popularidade no momento, a linguagem não é recente, tendo sido criada em 1995, por Brendan Eich e a equipe do navegador Netscape, ao verem a necessidade de uma linguagem de script para a web.

Figura 2 - Brendan Eich - Criador da linguagem de programação JavaScript



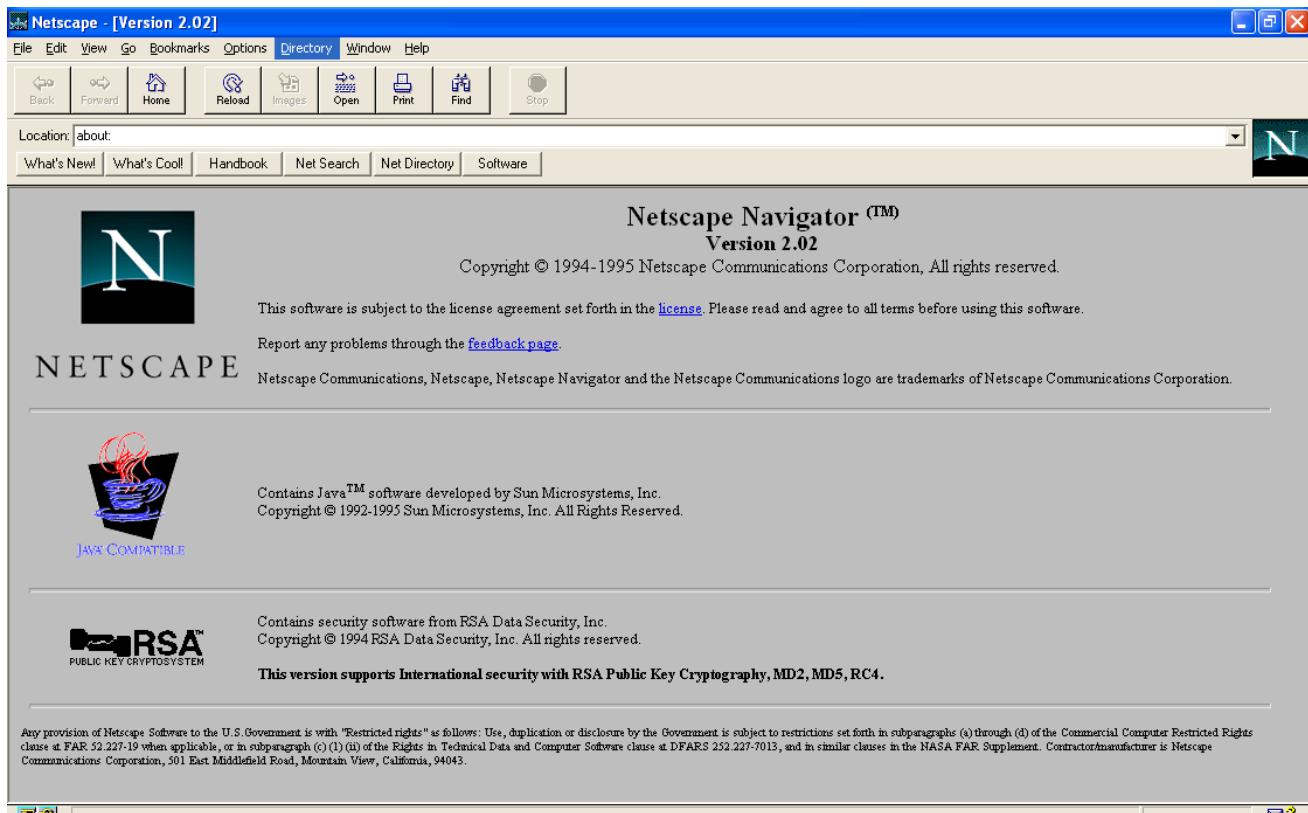
Fonte: Wikipedia

Curiosidade!

Você sabia que o projeto que deu início à linguagem JavaScript se chamava **Mocha**?

Para o lançamento da linguagem no navegador Netscape Navigator 2.0, a linguagem havia sido chamada de **LiveScript**. Entretanto, a partir de parceria estratégica com a Sun Microsystems (e considerando a popularidade da linguagem Java que havia sido lançada no mesmo ano), a linguagem foi renomeada para **JavaScript**, mesmo não havendo nenhuma ligação com a linguagem Java.

Figura 3 - Netscape Navigator 2.0, lançado em 1995.



Fonte: Wikipedia

Além desses nomes apresentados, você também pode conhecer a linguagem JavaScript por ECMAScript. Isso acontece pelo fato que, logo após o lançamento do JavaScript no navegador Netscape, diversas outras empresas da Web passaram a implementar sua própria versão de JavaScript, adicionando novos recursos. Foi o caso, por exemplo, da Microsoft, que lançou o Internet Explorer (IE) com sua própria linguagem de script chamada JScript.

Visando estabelecer um padrão à linguagem, a European Computer Manufacturers Association (ECMA) passou a atuar em uma padronização do JavaScript, criando assim a especificação **ECMAScript**, baseada principalmente na linguagem JavaScript, da Netscape e JScript, da Microsoft.

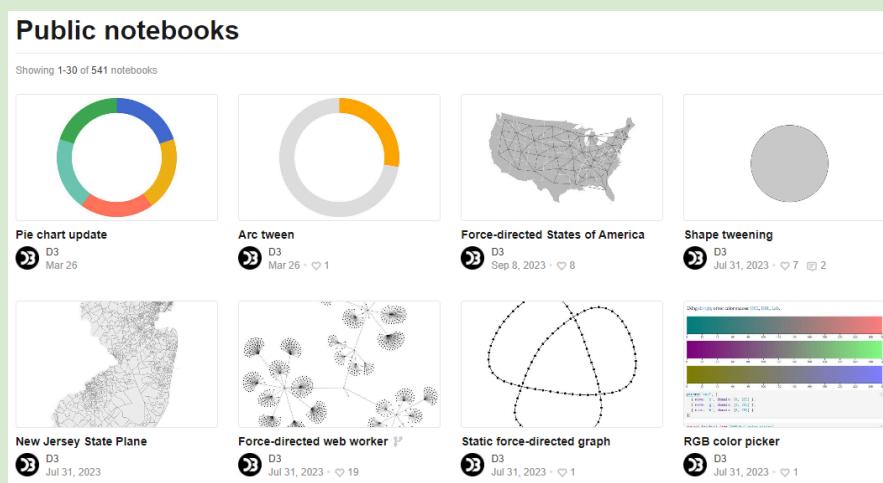
Até o momento a ECMA é a especificação principal sobre a qual JavaScript é baseada. A primeira edição do ECMAScript foi publicada em junho de 1997, seguida por várias revisões. A última edição é a ECMAScript 2024, que pode ser encontrada neste [link](#). Você também pode assistir [este vídeo](#), no qual o criador do JavaScript conta a história da linguagem.

Bem, após compreender um pouco mais sobre como surgiu a linguagem JavaScript, você deve estar se perguntando para quê ela é utilizada no momento. Confira as diversas funcionalidades que a linguagem oferece:

- **Manipulação do Document Object Model (DOM)**: JavaScript pode ser usada para acessar, modificar e manipular elementos HTML e CSS em tempo real, permitindo criar páginas web dinâmicas, como alterar o conteúdo, estilos, classes e atributos de elementos na página.
- **Resposta a Eventos**: com JavaScript você pode responder a eventos do usuário, como cliques e movimento do mouse.
- **Validação de Formulários**: você pode usar JavaScript para validar dados de entrada do usuário em formulários, antes de serem enviados para o servidor.
- **Requisições Ajax (Assíncronas)**: JavaScript permite fazer requisições assíncronas a servidores, permitindo a atualização de partes específicas de uma página sem recarregar a página inteira.
- **Armazenamento Local**: JavaScript permite usar recursos de armazenamento local, como localStorage para armazenar informações no navegador do usuário.
- **Animações e Efeitos**: JavaScript permite criar animações e efeitos visuais, como transições e animações de gráficos SVG.
- **Integração de APIs Externas**: JavaScript possibilita se integrar a APIs externas, como serviços de mapas, redes sociais e feeds de notícias.
- **Criação de Jogos Web**: JavaScript é usado para criar jogos web, desde jogos simples em 2D até experiências mais complexas e envolventes. Nesse contexto, existem até competições específicas para games criados via HTML, CSS e JavaScript, como o [js13kGames](#).

CURIOSIDADE!

Além das funcionalidades apresentadas, JavaScript também pode ser utilizado para visualização de dados. Veja um exemplo de uma biblioteca JavaScript para visualização dinâmica de dados, chamada D3.js por este [link](#).



Nesse [link](#) você pode encontrar também diversas animações criadas a partir de HMTL, CSS e JavaScript.

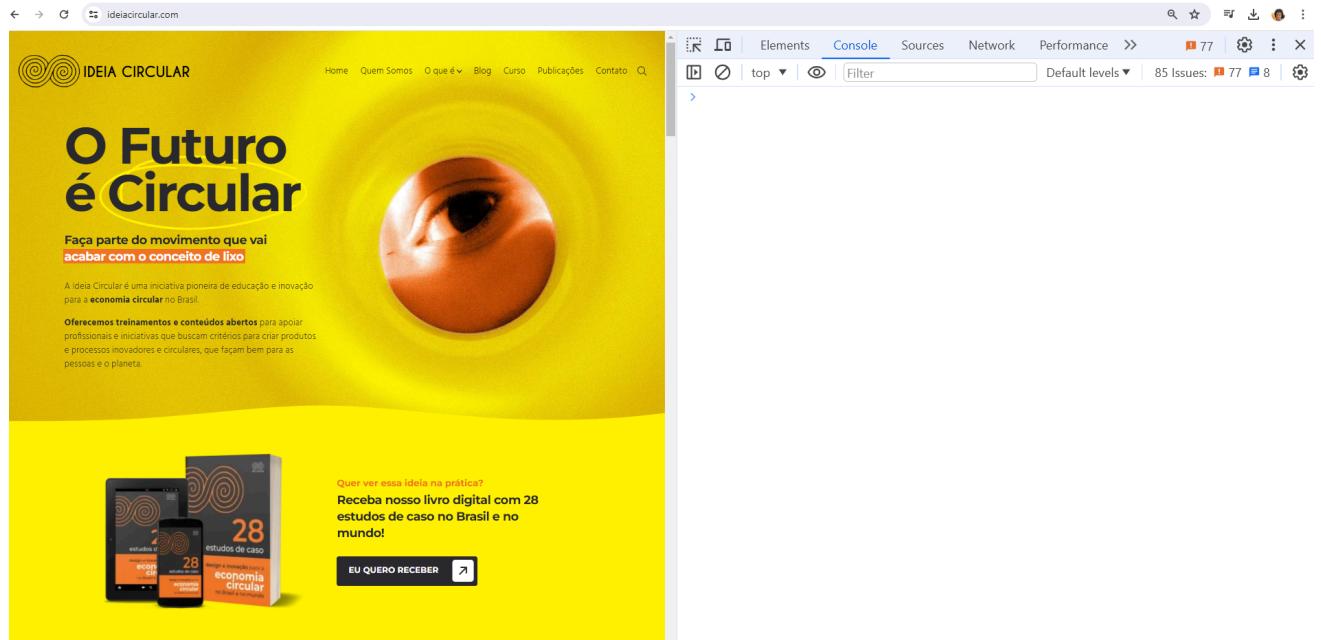
Passo 2 - Compreender as formas de uso da linguagem JavaScript

Antes de compreendermos mais a fundo as características da linguagem JavaScript, veremos algumas formas de utilizar essa linguagem.

Via console do navegador

Uma das formas para se utilizar o JavaScript é a partir do console do navegador. Essa ferramenta é utilizada para testar e entender o comportamento do seu código JavaScript. Ela está disponível nas ferramentas de desenvolvimento dos navegadores, podendo ser acessada clicando com o botão direito do mouse na página e selecionando "Inspecionar" e selecionando a aba "Console". Outra opção é acessar pelas teclas de atalho "Ctrl + Shift + J".

PRATICANDO: abra uma página web qualquer e clique com o botão direito do mouse na página, selecionando a opção Inspecionar. Após isso, acesse a ferramenta Console, conforme a figura a seguir. Ou use as teclas de atalho "Ctrl + Shift + J" para ir diretamente ao console.



Veja que foi aberto o console para a inserção de código JavaScript. Como exemplo, utilize o console inserindo o método `console.log()` para exibir mensagens no console do navegador, conforme exemplo a seguir:

23 Issues: ! 63 7

```
> const nome = "Rosangela";
  console.log("Olá, " + nome);
  Olá, Rosangela
< undefined
```

Para saber mais!

console.log refere-se a um método estático que apresenta uma mensagem de saída ao console. Porém, você sabia que existem outras variações que podem ser utilizadas? Além do log, você pode usar as seguintes opções:

```
console.log("mensagem")
console.info("mensagem")
console.debug("mensagem")
console.warn("mensagem")
console.error("mensagem")
```

Essas variações podem ser úteis para documentar ou apresentar algo que deseja em seu código. Por exemplo, ao usar a opção console.warn, a mensagem será apresentada em amarelo. Ao usar console.error, a mensagem será apresentada com fundo vermelho, conforme exemplos a seguir.

```
> console.warn("Atenção com esse comando")
```

⚠ ► Atenção com esse comando

```
< undefined
```

```
> console.error("Erro ao executar comando")
```

✖ ► Erro ao executar comando

Você pode também utilizar o console para executar comandos JavaScript diretamente no navegador.

Como exemplo, digite o código a seguir que emite um alerta ao usuário e pressione "Enter" para executá-lo, gerando o resultado conforme a figura a seguir.



23 Issues: ! 63 ⚬ 7

```
> alert("Olá, mundo!")
```

Essa ação utiliza o método alert para gerar a caixa de diálogo a seguir.

ideiacircular.com diz

Olá, mundo!

OK

Outro exemplo é utilizar o método prompt() para capturar uma entrada de dados de um usuário, conforme exemplo a seguir.

```
> var name = prompt('Informe seu nome:');
    alert('Olá, ' + name);
```

Esse código irá gerar o resultado a seguir.

Essa página diz

Informe seu nome:

Rosangela

OK

Cancelar

Essa página diz

Olá, Rosangela

OK

Caso queira executar novamente um comando, utilize a seta para cima do teclado, para navegar no histórico de comandos.

Incorporação direta em arquivo HTML

Uma outra opção possível é utilizar o código JavaScript em seu documento HTM. Isso pode ser feito utilizando a tag <script> dentro do arquivo HTML.

Você pode adicionar essas tags no <head> ou antes do fechamento do </body>, conforme exemplos a seguir.

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function saudacao() {
  alert("Olá, mundo!");
}
</script>
</head>
<body>
<button onclick="saudacao()">Clicar</button>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<button onclick="saudacao()">Clicar</button>
<script type="text/javascript">
function saudacao() {
  alert("Olá, mundo!");
}
</script>
</body>
</html>
```

Uso de arquivo externo

Uma boa prática também é a utilização de arquivo externo, com extensão .js, para incorporar seu código JavaScript. Você pode vincular esses arquivos externos ao documento HTML usando a tag <script>, conforme exemplo a seguir. Esse script pode estar dentro da tag <head> ou também na tag <body>.

script.js	index.html
<pre>function saudacao() { alert("Olá, mundo!"); }</pre>	<pre><!DOCTYPE html> <html> <head> <script src="script.js"></script></pre>

```
</head>
<body>
  <button onclick="saudacao()">Clicar</button>
</body>
</html>
```

Passo 3 - Conhecer as características da linguagem JavaScript

A partir do nome da linguagem podemos identificar que a linguagem JavaScript foi originalmente projetada para ser baseada em script, porém, atualmente ela se tornou uma linguagem de programação de propósito geral.

Vamos compreender algumas características da linguagem a seguir.

- Como vimos na história da linguagem, JavaScript é uma linguagem de programação projetada para a Web, possibilitando a construção de páginas web interativas, respondendo a ações do usuário e modificando o conteúdo da página de forma dinâmica.
- JavaScript é uma linguagem interpretada, de forma que o código-fonte é executado diretamente pelo navegador, sem a necessidade de compilação prévia.
- É considerada uma linguagem de alto nível e multiparadigma, suportando estilos como a programação orientada a objetos e a programação funcional.
- É possível incorporar o código JavaScript diretamente em seu HTML ou vinculá-lo a um arquivo externo.
- JavaScript é uma linguagem tipada dinamicamente, o que significa que você não precisa declarar o tipo de uma variável explicitamente. O tipo de uma variável é determinado em tempo de execução.
- Embora JavaScript seja executado nos navegadores do lado do cliente, a linguagem também pode ser executada do lado do servidor, por meio de tecnologias como o Node.js e a Apache CouchDB.
- Conforme veremos nas atividades posteriores, JavaScript pode ser usado para acessar e modificar o conteúdo de um documento HTML, permitindo a criação de páginas web dinâmicas mencionadas anteriormente.
- Além disso, JavaScript também pode ser usado para manipular estilos CSS, permitindo a alteração dinâmica da aparência de elementos HTML.

Por fim, é notável também que JavaScript possui atualmente uma ampla gama de bibliotecas e frameworks, como React, Angular e Vue.js, aumentando ainda mais a popularidade e funcionalidades da linguagem.

Além dessas características, veremos algumas outras características importantes a seguir.

Tipos de variáveis

Para declarar variáveis em JavaScript, podem ser utilizadas as palavras-chaves var, let ou const. No decorrer das aulas de JavaScript veremos mais informações sobre tais possibilidades, mas, em resumo: var possui escopo de função, let e const possuem escopo de bloco. Além disso, var e let permitem reatribuição, enquanto const declara variáveis imutáveis após a inicialização (constante).

Além disso, conforme mencionado, JavaScript é uma linguagem de **tipagem dinâmica**, ou seja, você não precisa declarar o tipo da variável explicitamente. Os tipos de variáveis em JavaScript incluem:

- Número (*number*): representa inteiros e números de ponto flutuante.
 - `var idade = 25;`
 - `const preco = 14.44;`
 - `let valor = 50;`
- String: para armazenar texto.
 - `let nome = "Joana";`
 - `let frase = 'Aprendendo JavaScript';`
- Boolean: verdadeiro (*true*) ou falso (*false*).
 - `var fraude = true;`
 - `let desligado = false;`
- Array: lista de valores, acessados por índices.
 - `let frutas = ["maçã", "uva", "laranja"];`
 - `let itens = [1, 2, 3, 4, 5]`
- Objeto: para armazenar propriedades e métodos.
 - `let pessoa = {
 nome: "Joana",
 idade: 44,
 cidade: "Londrina"
};`
- Função: para ser invocada para executar ações.
 - `function cumprimento(nome) {
 return `Olá, ${nome}!`;
}`
- null: ausência de qualquer valor ou objeto. Devido ao fato de JavaScript ser case-sensitive, null não é o mesmo que Null e NULL.
 - `let valorNulo = null;`
- Undefined: variável declarada, mas não inicializada.
 - `let valor;`

Comentários

Existem duas formas principais de adicionar comentários em JavaScript: comentários de uma linha e comentários de várias linhas, conforme exemplo a seguir.

```
// Comentário de uma linha.  
let idade = 42; // Comentário na mesma linha de código  
/*
```

comentário
de várias
linhas.

*/

Operadores aritméticos

JavaScript oferece diversos operadores aritméticos para realizar operações matemáticas em números.

- Adição (+)
- Subtração (-)
- Multiplicação (*)
- Divisão (/)
- Módulo (%)
- Incremento (++)
- Decremento (--)

Veja alguns exemplos com o uso de função:

```
function somaNumeros(num1, num2) {  
    return num1 + num2;  
}  
  
console.log(somaNumeros(10, 20));
```

```
function verificarParOuImpar(numero) {  
    if (numero % 2 === 0) {  
        return "Par";  
    } else {  
        return "Ímpar";  
    }  
}  
  
console.log(verificarParOuImpar(9));
```

Operadores de comparação

Os operadores de comparação são usados para comparar valores e expressões, geralmente resultando em um valor booleano, verdadeiro (true) ou falso (false). Veja exemplos a seguir:

- Igual (==): verifica se dois valores são iguais.
 - `5 == 5; // true`
 - `"10" == 10; // true`
- Estritamente Igual (===): verifica se dois valores são iguais e do mesmo tipo de dados.
 - `5 === 5; // true`

- "10" === 10; // false
- Diferente (!=): verifica se dois valores são diferentes.
 - 5 != 3; // true
 - "5" != 5; // false (conversão de tipo implícita)
- Estritamente Diferente (!==): verifica se dois valores são diferentes ou de tipos diferentes.
 - 5 !== "5"; // true
 - 5 !== 5; // false
- Maior que (>): verifica se o valor da esquerda é maior que o valor da direita.
 - 10 > 5; // true
- Menor que (<): verifica se o valor da esquerda é menor que o valor da direita.
 - 5 < 10; // true
- Maior ou Igual a (>=): verifica se o valor da esquerda é maior ou igual ao valor da direita.
 - 10 >= 5; // true
- Menor ou Igual a (<=): verifica se o valor da esquerda é menor ou igual ao valor da direita.
 - 5 <= 10; // true
- Operador Ternário (? :): operador de comparação que permite criar expressões condicionais compactas.
 - let idade = 20;
 - let resultado = idade >= 18 ? "Maior de Idade" : "Menor de Idade";

Operadores lógicos

Tais operadores são usados para combinar expressões booleanas anteriormente apresentadas. Existem três operadores lógicos principais: && (E lógico), || (OU lógico) e ! (NÃO lógico). Veja alguns exemplos:

Operador && - Retorna true se ambas as expressões forem verdadeiras.

```
const idade = 25;
const possuiCarteiraDeMotorista = true;

if (idade >= 18 && possuiCarteiraDeMotorista) {
  console.log("Pode dirigir.");
} else {
  console.log("Não pode dirigir.");
}
```

Operador || - Retorna true se pelo menos uma das expressões for verdadeira.

```
const temCartaoDeCredito = true;
const temDinheiro = false;

if (temCartaoDeCredito || temDinheiro) {
  console.log("Pode fazer a compra.");
```

```
} else {  
    console.log("Não pode fazer a compra.");  
}
```

Operador ! - Inverte o valor de uma expressão booleana

```
const usuarioLogado = true;  
  
if (!usuarioLogado) {  
    console.log("Faça o login para continuar.");  
} else {  
    console.log("Bem-vindo!");  
}
```

Estruturas de controle

Vimos nos exemplos anteriores o uso do if, como estrutura de controle condicional. Além dessas opções, também podem ser utilizadas os seguintes operadores de controle:

else-if - Permite verificar múltiplas condições em sequência.

```
const diaDaSemana = "terça";  
  
if (diaDaSemana === "sábado" || diaDaSemana === "domingo") {  
    console.log("É fim de semana!");  
} else if (diaDaSemana === "segunda" || diaDaSemana === "terça") {  
    console.log("É início da semana.");  
} else {  
    console.log("É um dia comum.");  
}
```

while - Usado para criar loops enquanto uma condição for verdadeira.

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

do...while - Garante que o bloco de código seja executado pelo menos uma vez, mesmo se a condição for falsa.

```
let numero = 1;  
  
do {  
    console.log("Número: " + numero);  
    numero++;  
}
```

```
} while (numero <= 3);
```

for - Usado para criar loops com um contador.

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

for...in - Alternativa para criar loop. Executa iterações a partir de uma variável informada, permitindo percorrer as propriedades de um objeto.

```
var produtos = [  
    { id: 1, tipo: 'camiseta', quantidade: 20 },  
    { id: 2, tipo: 'livro', quantidade: 15 }  
]  
  
var qtdtotal = 0  
  
for(const i in produtos) {  
    qtdtotal += produtos[i].quantidade  
}
```

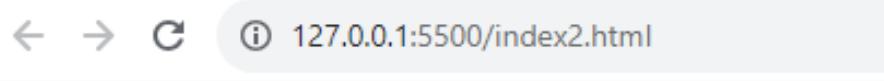
switch - Usado para fazer múltiplas verificações em uma única variável.

```
let fruta = "maçã";  
  
switch (fruta) {  
    case "maçã":  
        console.log("É uma maçã.");  
        break;  
    case "banana":  
        console.log("É uma uva.");  
        break;  
    default:  
        console.log("Não está na lista.");  
}
```

PRATICANDO: Para dar sequência ao conteúdo de JavaScript, crie um arquivo index.html e insira o código a seguir, visualizando o resultado a partir do Live Server. Nesse exemplo, foi utilizado document.write(), um método em JavaScript que permite escrever conteúdo diretamente no documento HTML a partir do código JS. Posteriormente veremos outras alternativas mais eficientes para essa ação.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aprendendo Javascript</title>
</head>
<body>
    <script type="text/javascript">
        var i=0;
        for (i=0;i<=8;i++){
            document.write("Número digitado: " + i);
            document.write("<br />");
        }
    </script>
</body>
</html>
```

Ao abrir o navegador, o resultado deve ser similar ao exemplo a seguir:



Número digitado: 0
Número digitado: 1
Número digitado: 2
Número digitado: 3
Número digitado: 4
Número digitado: 5
Número digitado: 6
Número digitado: 7
Número digitado: 8

Objeto Date

O objeto Date é usado em JavaScript para trabalhar com datas e horários. Veja alguns dos recursos a partir do exemplo a seguir:

```
const dataAtual = new Date();

const dia = dataAtual.getDate(); // Dia do mês (1-31)
const mes = dataAtual.getMonth(); // Mês (0-11, janeiro é 0)
const ano = dataAtual.getFullYear();
const hora = dataAtual.getHours(); // Hora (0-23)
const minutos = dataAtual.getMinutes(); // Minutos (0-59)
const segundos = dataAtual.getSeconds();

alert(`Data atual: ${dia}/${mes + 1}/${ano}`);

console.log(`Horário atual: ${hora}:${minutos}:${segundos}`);
```

Nesse exemplo, o conteúdo dentro de `alert()` foi implementado usando o conceito chamado “Template strings”, no qual o conteúdo é envolvido por ` ` em vez de aspas simples ou duplas. Esse recurso permite utilizar *placeholders* que são indicados por um círculo seguido de chaves (exemplo: `${dia}`)

Ao visualizar o resultado, deve ser mostrado um alerta na página com a data atual e uma mensagem no console com a hora atual, conforme exemplo a seguir.

Essa página diz

Data atual: 8/5/2024

OK

`console.log(`Horário atual: ${hora}:${minutos}:${segundos}`);`

Horário atual: 11:9:36

[VM12:12](#)

Você também pode aprender sobre as outras funções do Objeto Date por meio desse [link do MDN](#).

Considerações finais

Caso tenha chegado até aqui, você conseguiu completar o conteúdo do primeiro tutorial sobre JavaScript. A partir desses recursos, você passa a compreender a base para o desenvolvimento de funcionalidades via JavaScript. Nas aulas seguintes veremos ainda mais funcionalidades para tornar as páginas ainda mais dinâmicas.

Bom estudo!