



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](#)
4. [Atividade prática com 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 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 a décima segunda em 2014.

Figura 1 - Índice das linguagens de programação - Outubro de 2023

Oct 2023	Oct 2022	Change	Programming Language	Ratings	Change
1	1		 Python	14.82%	-2.25%
2	2		 C	12.08%	-3.13%
3	4	▲	 C++	10.67%	+0.74%
4	3	▼	 Java	8.92%	-3.92%
5	5		 C#	7.71%	+3.29%
6	7	▲	 JavaScript	2.91%	+0.17%
7	6	▼	 Visual Basic	2.13%	-1.82%

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

Embora JavaScript esteja em alta popularidade no momento, a linguagem não é tão 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.

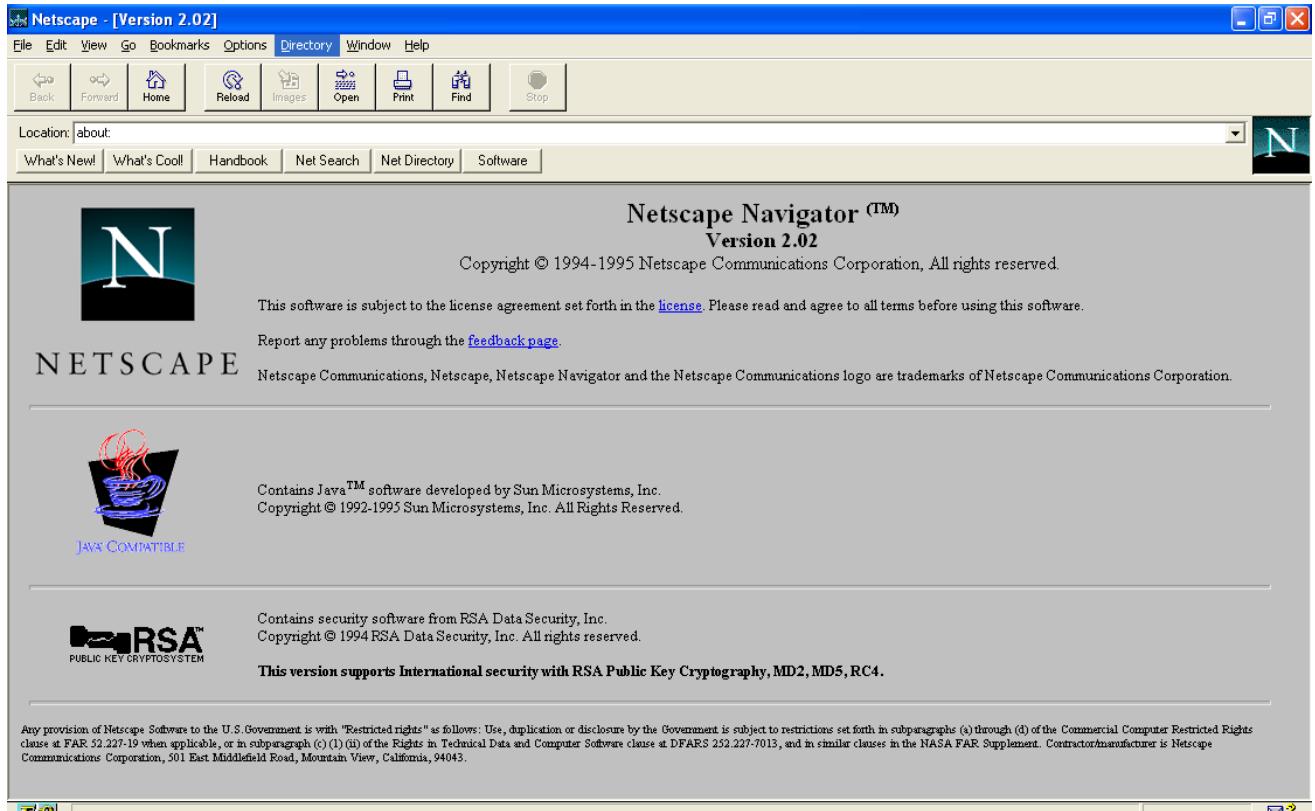
Figura 2 - Brendan Eich - Criador da linguagem de programação JavaScript



Fonte: Wikipedia

Curiosidade: 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 originalmente 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 ligação com a linguagem Java.

Figura 3 - Netscape Navigator 2, lançado em 1995



Fonte: Wikipedia

Além desses nomes, 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 o ECMA é a especificação principal sobre a qual o JavaScript é baseado. Sua primeira edição do ECMAScript foi publicada em junho de 1997, seguida por várias revisões. A última edição é a ECMAScript 2023, 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. Veja as diversas funcionalidades que a linguagem oferece:

- **Manipulação do DOM (Document Object Model):** 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.

Veja um exemplo de uma biblioteca JavaScript para visualização dinâmica de dados, chamada D3.js por este [link](#).

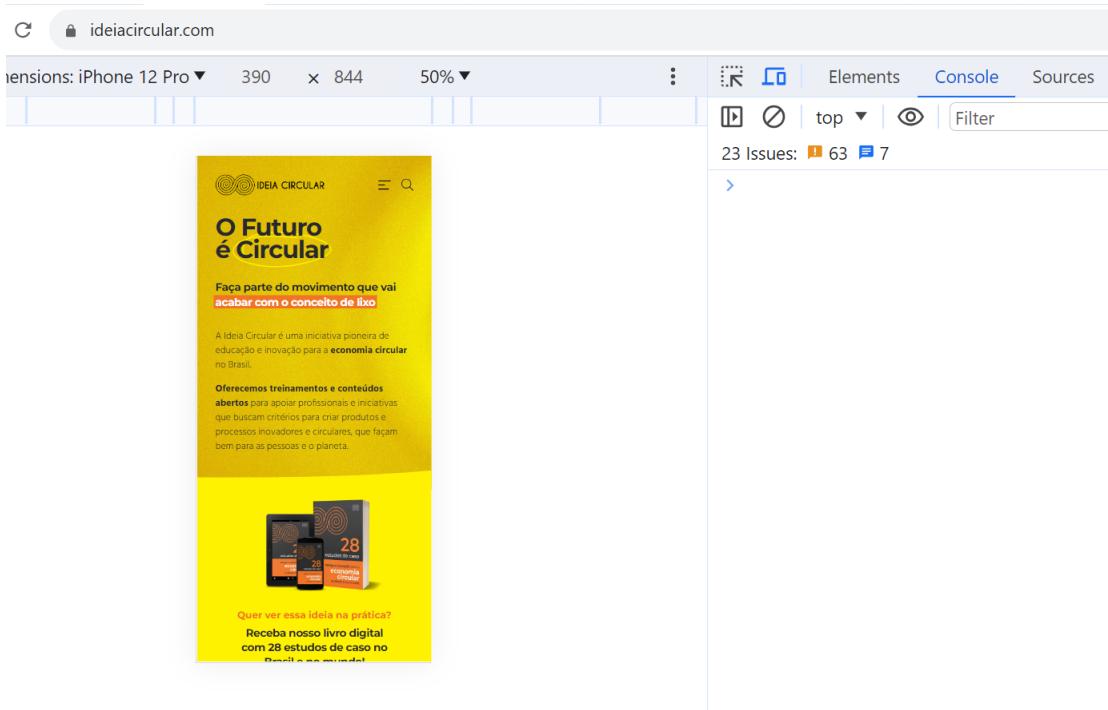
Passo 2 - Compreender as formas de uso da linguagem JavaScript

Antes de compreendermos mais características da linguagem JavaScript, veremos algumas formas de utilizar a 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".

PRATICANDO: abra uma página web 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.



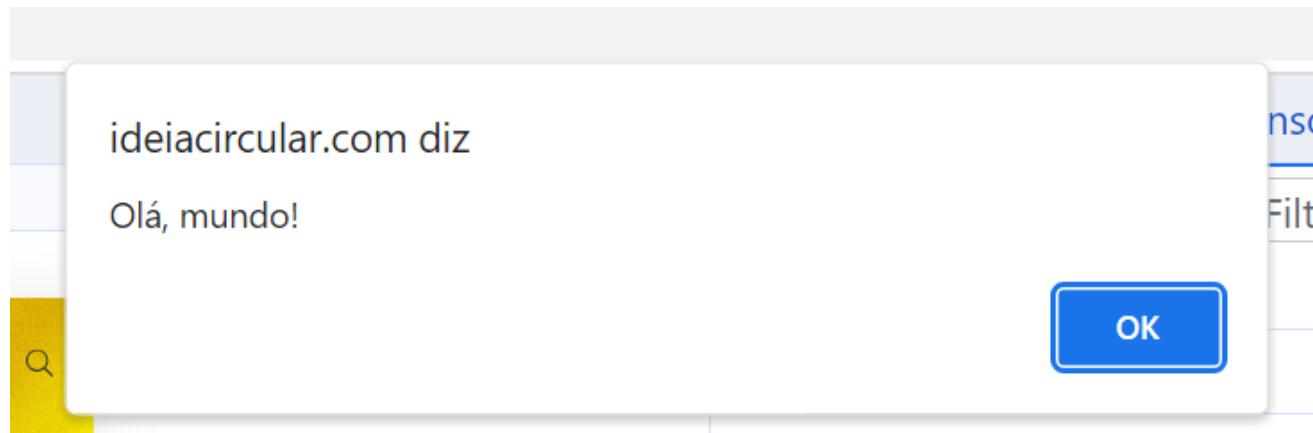
Utilize a ferramenta inserindo o método `console.log()` para exibir mensagens no console do navegador, conforme exemplo a seguir:

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. At the top, there are icons for back, forward, and search, followed by tabs for 'Elements', 'Console' (which is blue and underlined), 'Sources', 'Network', and a notifications icon showing '63'. Below the tabs is a toolbar with a play button, a stop button, 'top', a refresh eye icon, a 'Filter' input field, and a 'Default levels' dropdown. A message '23 Issues: ! 63 ⚡ 7' is displayed. The main area shows a code snippet and its output:
> const nome = "Rosangela";
console.log("Olá, " + nome);
Olá, Rosangela
< undefined

Além disso, você pode também utilizar o console para executar comandos JavaScript diretamente. Como exemplo, digite o código a seguir e pressione "Enter" para executá-lo, gerando o resultado conforme a figura a seguir.

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The toolbar and message bar are identical to the previous screenshot. The main area shows a code snippet:
> alert("Olá, mundo!")

Essa ação utiliza a função `alert` para gerar a caixa de diálogo a seguir.



Incorporação direta em arquivo HTML

Uma outra opção é utilizar o código JavaScript em seu documento HTML. 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 exemplo 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.

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

Passo 3 - Conhecer as características da linguagem JavaScript

Já pelo nome da linguagem podemos identificar que JavaScript foi originalmente projetado para ser usada como linguagem de script, porém, atualmente ela se tornou uma linguagem de programação de propósito geral.

Mas 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.
- 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 tem escopo de função, let e const têm escopo de bloco. Além disso, var e let permitem reatribuição, enquanto const declara variáveis imutáveis após a inicialização.

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.
 - let valorNulo = null;
- Undefined: variável declarada, mas não inicializada.
 - let indefinido;

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 (--)
- Operadores de Atribuição Composta

Veja alguns exemplos com o uso de função:

```
function somaNumeros(num1, num2) {  
    return num1 + num2;  
}  
  
console.log(somaNumeros(5, 3));
```

```
function verificarParOuImpar(numero) {  
    if (numero % 2 === 0) {  
        return "Par";  
    } else {  
        return "Ímpar";  
    }  
}  
  
console.log(verificarParOuImpar(7));
```

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);
}
```

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: crie um arquivo index.html e insira o código a seguir, visualizando o resultado a partir do Live Server.

```
<!DOCTYPE html>
<html lang="en">
<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>
```

Nesse exemplo, foi utilizado `document.write()`, um método em JavaScript que permite escrever conteúdo diretamente no documento HTML a partir do código JavaScript.

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}`);
```

Passo 4 - Atividade prática com JavaScript

Para validar o conhecimento visto nos tópicos anteriores, realize as atividades a seguir, enviando-as no moodle da disciplina.

1 - Escreva um programa em JavaScript (documento interno) que faça a iteração de 1 a 10. Para cada iteração, deve ser verificado se o número atual é par ou ímpar, exibindo uma mensagem no console (função `console.log()`), similar à imagem a seguir:

```
1 = ímpar  
2 = par  
3 = ímpar  
4 = par  
5 = ímpar  
6 = par  
7 = ímpar  
8 = par  
9 = ímpar  
10 = par
```

OBSERVAÇÃO: A saída da função `console.log()` é exibida no console do navegador. Para visualizar a saída, siga os seguintes passos:

- Abra no navegador a página HTML que contém o código JavaScript que você implementou.
- Pressione a tecla F12 para abrir as ferramentas de desenvolvedor do navegador ou clique em inspecionar página com o botão direito do mouse.
- Clique na guia "Console" nas ferramentas de desenvolvedor.
- Veja a saída do código nessa aba.
- Caso queira, você pode executar um novo código JavaScript que contém a função `console.log()` nessa aba.

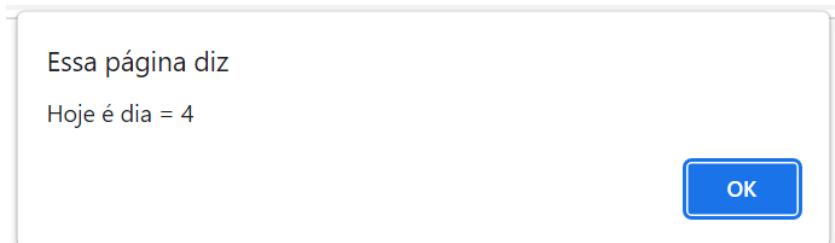
2. Refaça a atividade anterior, porém, utilize um arquivo javascript externo e, ao invés de apresentar a mensagem no console, apresente em uma tag h3, similar à imagem a seguir:

```
1 = ímpar  
2 = par  
3 = ímpar  
4 = par  
5 = ímpar  
6 = par  
7 = ímpar  
8 = par  
9 = ímpar  
10 = par
```

3. Faça um programa em JavaScript (documento interno) que faça a iteração de 1 a 10. Para cada iteração, deve ser apresentado o valor do índice ao quadrado, exibindo o valor em uma lista não ordenada, similar à imagem a seguir:

- 1
- 4
- 9
- 16
- 25
- 36
- 49
- 64
- 81
- 100

4. Faça um programa em JavaScript (documento interno) que apresente um alerta com a mensagem referente ao dia atual (utilize o objeto Date para capturar o dia), conforme imagem a seguir:



5. Faça um programa em JavaScript (documento interno) que crie um array contendo os 4 primeiros objetivos de desenvolvimento sustentável (ODS). Apresente cada ODS em uma tag h2 e no console, conforme imagens a seguir.

ODS 1 - Erradicação da Pobreza

ODS 2 - Fome Zero e Agricultura Sustentável

ODS 3 - Saúde e Bem-Estar

ODS 4 - Educação de Qualidade

ODS 1 - Erradicação da Pobreza

ODS 2 - Fome Zero e Agricultura Sustentável

ODS 3 - Saúde e Bem-Estar

ODS 4 - Educação de Qualidade

Entregar um arquivo .zip com os códigos realizados em cada atividade.

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!