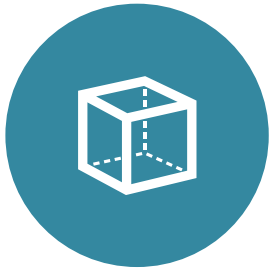
PROGRAMAÇÃO WEB II

Prof. Esp. José Olinda

OBJETIVOS

DESENVOLVER SISTEMAS PARA WEB COM PÁGINAS DINÂMICAS CONSTRUÍDAS NO LADO DO CLIENTE



APLICAR O PARADIGMA DE ORIENTAÇÃO A OBJETOS EM SISTEMAS WEB

APRENDER TÉCNICAS PARA DINAMIZAÇÃO E INTERATIVIDADE EM PÁGINAS WEB

DESENVOLVER APLICAÇÕES WEB ESTRUTURADAS SEGUNDO O PADRÃO MVC

Unidade I

INTRODUÇÃO AO JAVASCRIPT

Características, Ambiente de Desenvolvimento, Sintaxe Básica

O QUE É JAVASCRIPT

- JavaScript é uma linguagem de programação orientada a objetos empregada pela maioria dos sites junto com HTML e CSS para criar experiências de usuário robustas, dinâmicas e interativas.
- Foi introduzida em 1995 e desde então se tornou uma das mais populares, com suporte para todos os principais navegadores da web.
- Os programas JavaScript são usados tanto no lado do cliente quanto no lado do servidor para adicionar funcionalidade às páginas da web.
- Uma pesquisa de desenvolvedor **Stack Overflow** de 2016 listou JavaScript como o desenvolvedor front-end e tecnologia de desenvolvimento back-end mais popular.

Fonte: https://www.edx.org/learn/javascript

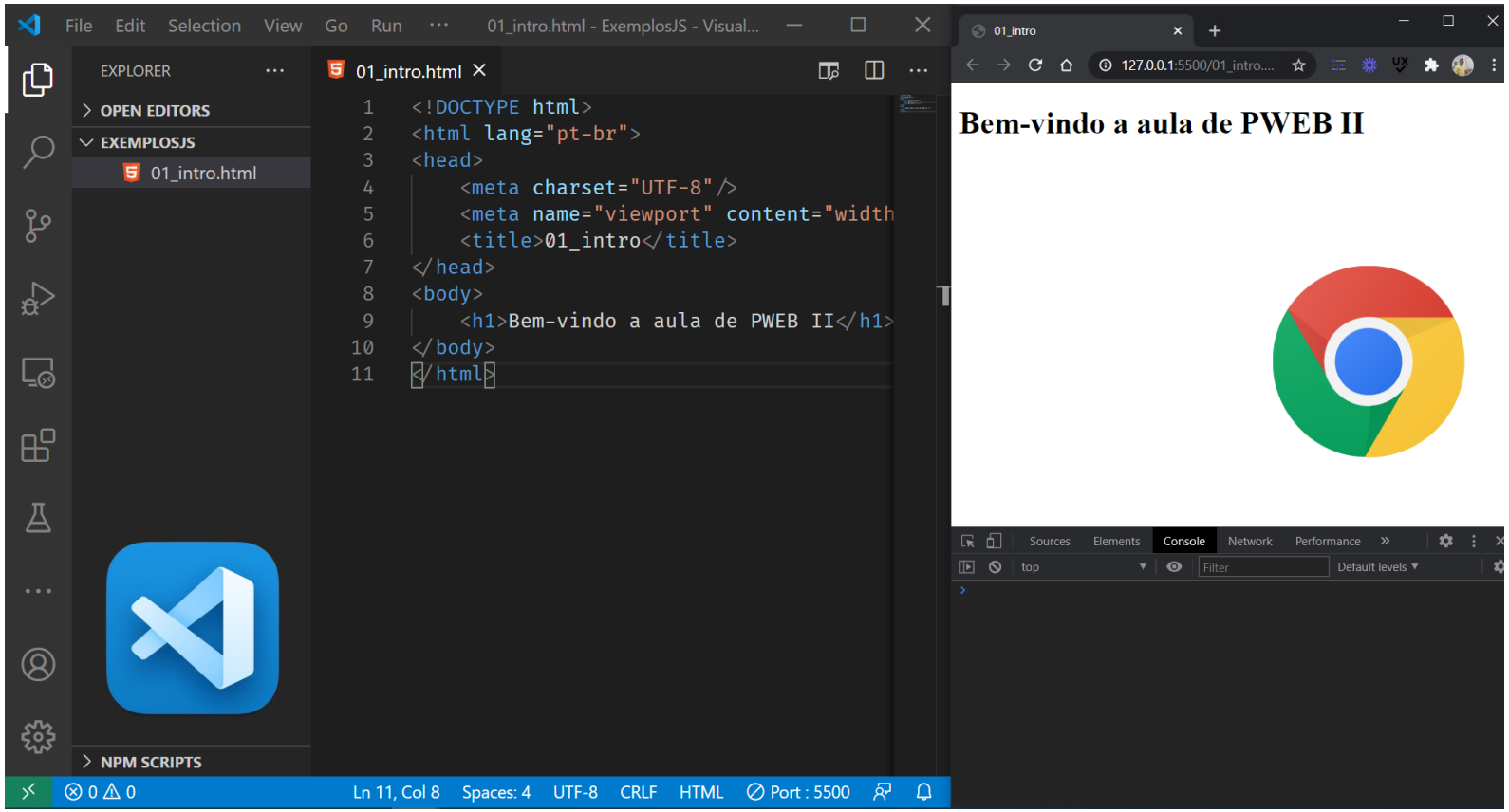
POR QUE APRENDER JAVASCRIPT?

- JavaScript é uma das principais linguagens de programação utilizadas no desenvolvimento web.
- O JavaScript não é apenas fácil de usar e muito versátil, mas aqueles que têm as habilidades para usá-lo são muito procurados.
- As empresas estão sempre procurando pessoas com proficiência em JavaScript.
- É uma ótima linguagem para aprender se você estiver interessado em desenvolvimento web.

Fonte: https://www.edx.org/learn/javascript

AMBIENTE DE DESENVOLVIMENTO JS

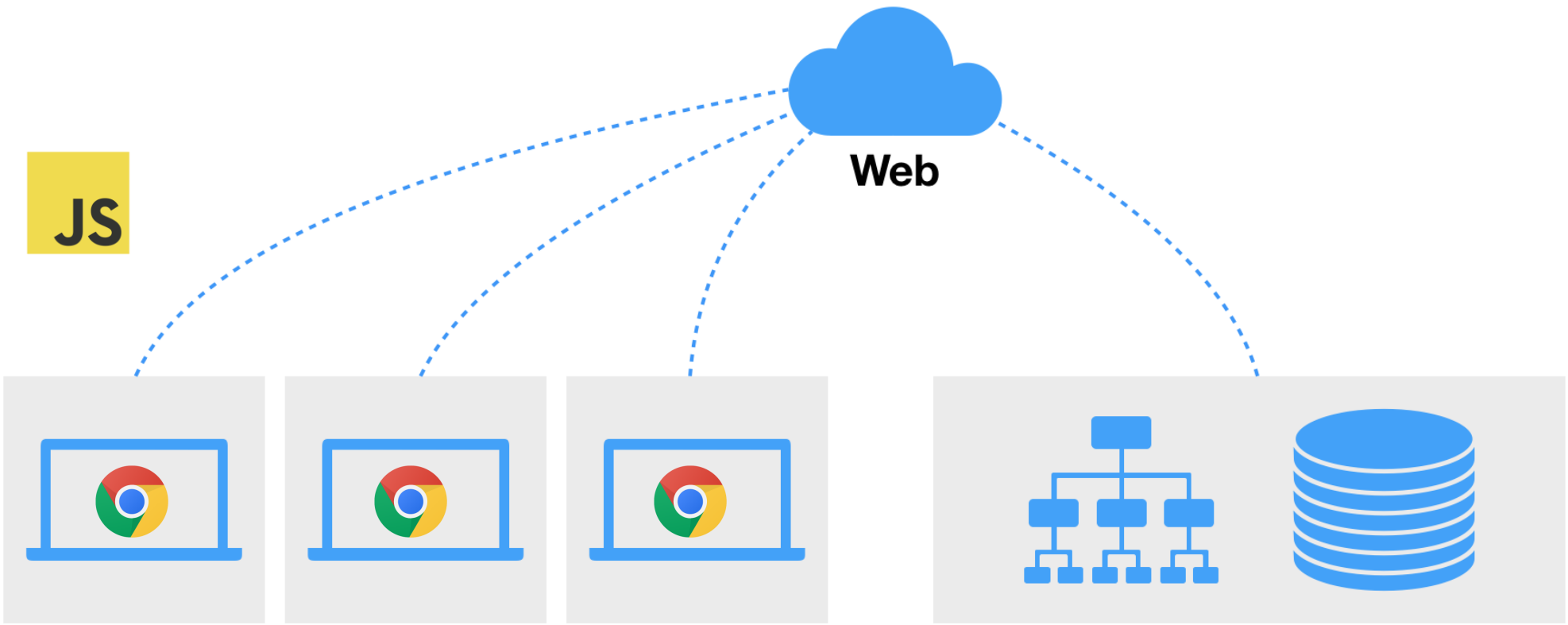
- Editor de texto
  - VS Code, Sublime, Atom
- Navegador
  - Chrome, Firefox, Edge
- Console do navegador



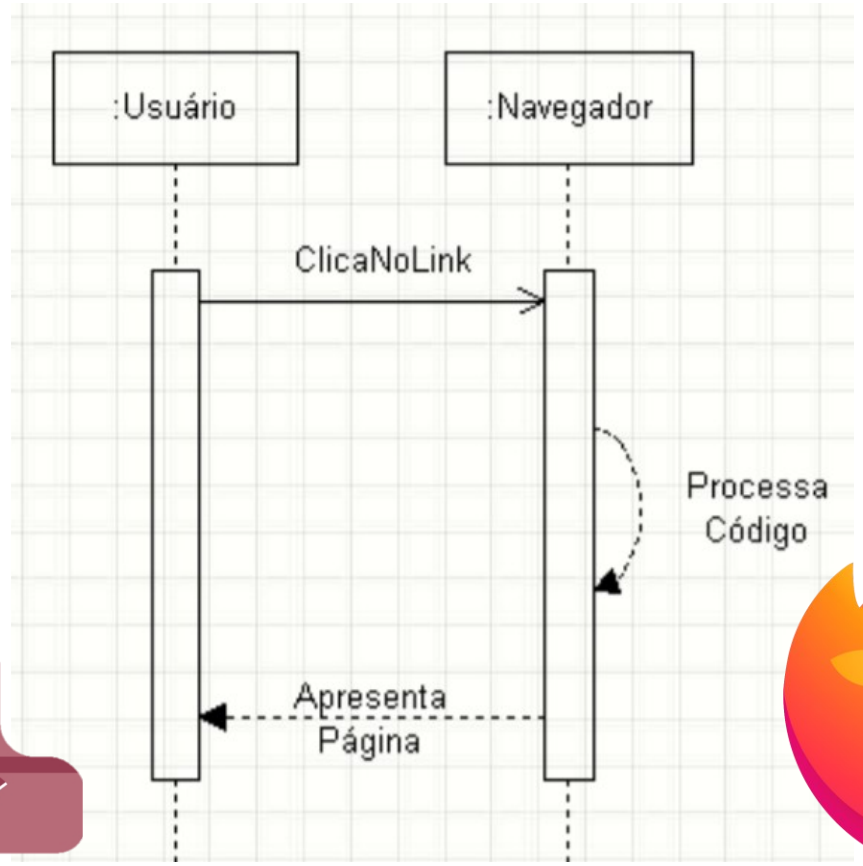
FUNÇÕES DO JAVASCRIPT

- O JavaScript é capaz de:
- Modificar conteúdo (texto e HTML)
- Modificar Apresentação (CSS)
- Inserir, remover e editar imagens
- Interagir com formulários
- Atualizar a página dinamicamente (sem recarregar)
- Manipular a janela do navegador

CLIENTE X SERVIDOR



PROCESSAMENTO NO LADO DO CLIENTE



- Exemplo de processamento no lado do cliente

ADICIONANDO CÓDIGO JS A SUA PÁGINA

- Interno
  - Use a tag *script* para adicionar código JS a suas páginas HTML. O código do programa JavaScript ficará no mesmo arquivo da sua página HTML, delimitado pela tag *script*.
- Externo
  - Use um arquivo externo para escrever seus programas JS. O programa é criado em um arquivo externo com extensão *.js* e adicionado a página HTML usando a tag *script* com o atributo *src* informando onde está localizado o arquivo *js.*
- Inline
  - Os elementos HTML possuem atributos relacionados aos eventos (clicar, modificar, receber foco) que podem receber código JavaScript diretamente. Este código será executado sempre que o evento relacionado for executado/disparado pelo navegador, a partir de interações do usuário.

ADICIONANDO CÓDIGO JS A SUA PÁGINA

- Interno

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meu primeiro programa</title>
  </head>
  <body>
    <h2>Meu primeiro programa em javaScript</h2>
    <script>
      alert("Olá, Mundo!");
    </script>
  </body>
</html>
```

ADICIONANDO CÓDIGO JS A SUA PÁGINA

- Externo

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meu primeiro programa</title>
  </head>
  <body>
    <h2>Este script é externo</h2>
    <script src=“ex2.js”></script>
  </body>
</html>
```

```
alert("Olá, Mundo!");
```

Um arquivo externo armazena o programa em JavaScript. Esta abordagem facilita a manutenção e a separação do código.

ADICIONANDO CÓDIGO JS A SUA PÁGINA

- Inline

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meu primeiro programa</title>
  </head>
  <body>
    <h2>Outro programa javaScript</h2><br>
    <button
       onclick="alert('Você clicou no botão')">
       Clica aqui
    </button>
  </body>
</html>
```

HEAD X BODY

- A tag *script* pode ser usada na seção *head* ou na seção *body.* Entretanto, é fortemente recomendado que seja utilizada no final da tag *body* (antes do fechamento).
- Quando inserido na tag *head*, o *script* será interpretado antes dos demais elementos da página serem interpretados. Isso deixará o seu site mais lento.
- Quando adicionado ao final da tag *body*, todos os elementos da página já estarão carregado antes do processamento do *script*. Isso evita que o JavaScript cause uma obstrução no carregamento da página.

POSSIBILIDADES DE EXIBIÇÃO COM JAVASCRIPT

JavaScript pode "exibir" dados de maneiras diferentes:

- Escrevendo em um elemento HTML, usando **`innerHTML`**.
- Escrevendo na saída HTML usando **`document.write()`**.
- Escrevendo em uma caixa de alerta, usando **`window.alert()`**.
- Escrevendo no console do navegador, usando **`console.log()`**.

innerHTML

- Para acessar um elemento HTML, podemos usar o método **document.getElementById(*id*)**.
- O atributo **id** define o elemento a partir da propriedade id do elemento HTML.
- A propriedade **innerHTML** permite retornar ou alterar o conteúdo HTML do elemento selecionado.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Exibindo conteúdo com JS</h1>
    <p>Usando o método innerHTML</p>
    <p id="paragrafo"></p>

    <script>
        document.getElementById("paragrafo").innerHTML = “Novo parágrafo";
    </script>
  </body>
</html>
```

document.write()

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Exibindo conteúdo com JS</h1>
    <p>Inserir conteúdo usando document.write().</p>

    <script>
      document.write(3 + 12);
    </script>
  </body>
</html>
```

- Para fins de teste, é conveniente usar **`document.write()`**
- Usar **`document.write()`** após o carregamento de um documento HTML excluirá todos os HTML existentes.
- A propriedade **`innerHTML`** é forma ideal para adição de conteúdo HTML a sua página. Evite usar **`document.write()`**

QUAL O RESULTADO DESTE CÓDIGO?

- Execute o exemplo a seguir e veja o resultado.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Exibindo conteúdo com JS</h1>
    <p>Inserir conteúdo usando document.write().</p>

    <button type="button" onclick="document.write('Você clicou no botão')">
      Clique aqui
    </button>

  </body>
</html>
```

alert()

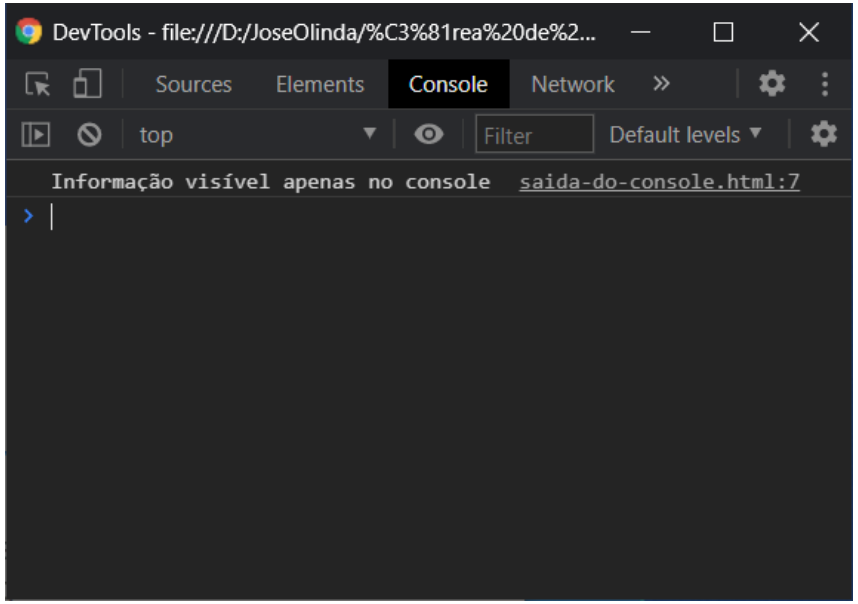
- Você pode usar uma caixa de alerta para exibir dados.
- Em JavaScript, o objeto de **window** é o objeto de escopo global, o que significa que variáveis, propriedades e métodos por padrão pertencem ao objeto **window**.
- Portanto, usar window para acessar métodos e propriedades globais é opcional.
- **alert()** é muito obstrutivo, impedindo completamente a interação com a página até que a janela de alerta seja encerrada. Use moderadamente.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Exibindo conteúdo com JS</h1>
    <p>Usando o método window.alert()</p>

    <script>
      window.alert("Você não vai conseguir usar a página até clicar em Ok.");
      alert("Posso usar o alert sem o window");
    </script>
  </body>
</html>
```

`console.log()`

- Para fins de depuração, você pode chamar o método **`console.log()`** para exibir os dados no console do navegador.
- As informações exibidas em **`console.log()` não aparecem na sua página web**. Isso é muito útil para os desenvolvedores, mas significa que **não deverá ser usada para exibir informações ao usuário**.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Exibindo conteúdo com JS</h1>
    <p>Usando o método window.alert()</p>

    <script>
      console.log("Informação visível apenas no console");
    </script>
  </body>
</html>
```



INSTRUÇÕES (STATEMENTS)

- Cada linha de código em JavaScript é chamada de instrução;
- ***Uma instrução é finalizada usando ponto-e-vírgula.***
- As instruções JavaScript são compostas por:
  - Valores
  - Operadores
  - Expressões
  - Palavras-chave
  - Comentários

```
document.getElementById("app").innerHTML = “Bom dia";
var a = 5; var b = 6; var c = a + b;
var person = "Jose";
var person="Jose";
```

SINTAXE BÁSICA

- Pega emprestado a maior parte de sua sintaxe do Java, mas também é influenciado por Awk, Perl e Python.
- É case-sensitive;
- É fracamente (ou dinamicamente) tipada. Ou seja, os tipos de dados das variáveis e objetos são definidos em tempo de execução.

PALAVRAS RESERVADAS

Palavra-chave	Descrição
break	Termina um switch ou um loop
continue	Pula de um loop e começa no topo
debugger	Para a execução de JavaScript e chama (se disponível) a função de depuração
do ... while	Executa um bloco de instruções e repete o bloco, enquanto uma condição for verdadeira
for	Marca um bloco de instruções a serem executadas, desde que uma condição seja verdadeira
function	Declara uma função
if ... else	Marca um bloco de instruções a serem executadas, dependendo de uma condição
return	Sai de uma função e retorna um valor ou expressão
switch	Marca um bloco de instruções a serem executadas, dependendo dos casos diferentes
try ... catch	Implementa o tratamento de erros em um bloco de declarações
var, let	Declara uma variável
const	Declara uma constante