SI401

Programação para a Web

2º Semestre - 2024

Programação *Front-End*: *JavaScript*
Unidade 09

Prof. Guilherme Palermo Coelho
guilherme@ft.unicamp.br

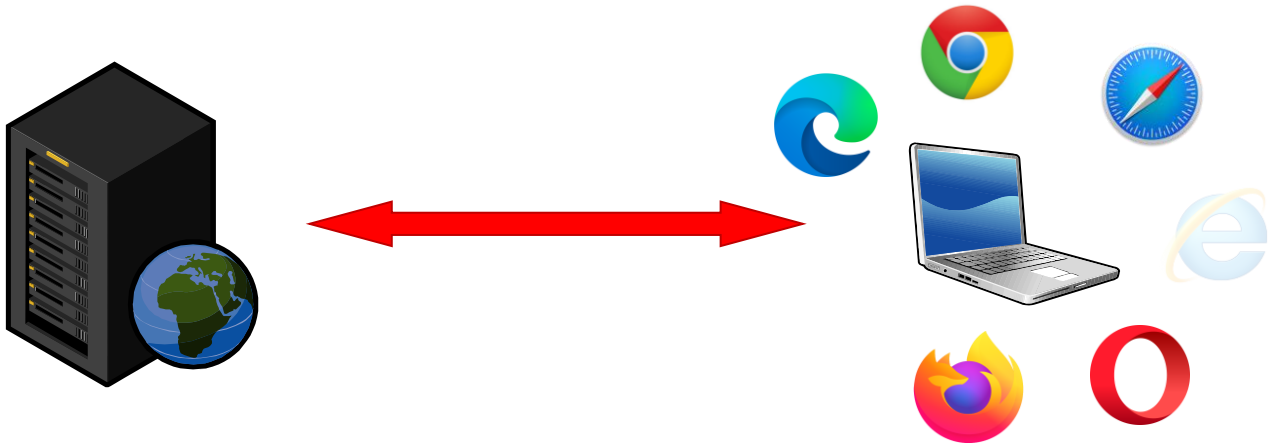
Roteiro

- O *Front-End*;
- O que é *JavaScript*?
- O que se pode fazer com *JavaScript*?
- Um primeiro exemplo de *JavaScript*;
- O objeto DOM;
- Onde colocar scripts em um documento?
- Instruções e Funções em *JavaScript*;
- Referências.

Front-End

O que é *Font-End*?

- **Programação web**: pode ser entendida como o desenvolvimento de software baseados na estrutura da web existente atualmente.
  - **Nessa disciplina**: software baseado em *scripts* executados tanto do lado do cliente quanto do servidor.



Back-End: camada “interna” do sistema

Front-End: HTML/CSS/*JavaScript*

O QUE É *JAVASCRIPT*?

O que é *JavaScript*?

- *JavaScript* é uma *linguagem leve* de programação (linguagem de *script*), desenvolvida com o intuito de adicionar ***interatividade*** a páginas Web;
- É uma linguagem ***orientada a objetos***;
- Os *scripts* produzidos em *JavaScript* geralmente são incorporados ***diretamente*** nos documentos HTML;
  - Isto no contexto de ***browser scripting***, que será considerado aqui.
- *JavaScript* é uma linguagem ***interpretada***.

O que é *JavaScript*?

- Foi inventada pela Netscape e surgiu com o navegador Netscape 2.0:
  - Desde 1996 passou a ser incorporada em todos os navegadores;
  - Em 1997 foi padronizada pela ECMA (ECMA-262);
    - Última edição (15ª): Junho, 2024.
  - Em 1998 foi padronizada pela ISO (ISO/IEC 16262).
- *JavaScript* ***não*** *é Java!*
  - Apenas possui semelhanças de sintaxe e convenções de nome.

O QUE SE PODE FAZER COM *JAVASCRIPT*?

O que se pode fazer com *JavaScript*?

- *JavaScript* provê uma ferramenta de ***programação*** para desenvolvedores Web:
  - Possui uma sintaxe simples;
- *JavaScript* é capaz de ***reagir a eventos***;
  - Códigos em *JavaScript* podem ser executados quando "**algo ocorre**":
    - Quando uma página termina de carregar;
    - Quando o usuário clica em determinado elemento;
    - ...

O que se pode fazer com *JavaScript*?

- *JavaScript* é capaz de ***ler*** e ***modificar*** o conteúdo de elementos HTML;
- *JavaScript* pode ser usada para ***validar*** dados antes deles serem enviados ao servidor;
- *JavaScript* pode ser usada para ***detectar*** qual o navegador do usuário;
- *JavaScript* permite o armazenamento e a recuperação de informações do computador do usuário;
- ...

UM PRIMEIRO EXEMPLO DE *JAVASCRIPT*

Um primeiro exemplo de *JavaScript*

```
<html>
<body>

<h1>Primeiro exemplo de <i>Javascript</i></h1>

<p id="demo">Um parágrafo.</p>

<script type="text/javascript">
document.getElementById("demo").innerHTML=Date();
</script>

</body>
</html>
```

Primeiro exemplo de *Javascript*

Fri Mar 16 2012 15:11:30 GMT-0300 (E. South America Standard Time)

- Códigos em *JavaScript* são inseridos em documentos HTML através da *tag* **<script>**:
  - O atributo "**type**" com o valor "**text/javascript**", **não** é necessário em versões atuais de HTML.

Um primeiro exemplo de *JavaScript*

- No exemplo anterior, as *tags* **<script> </script>** indicam onde começa e onde termina o trecho de código em *JavaScript*;
  - O navegador é responsável por interpretar e executar este código;
  - Sem as *tags* **<script>** e **</script>**, o navegador interpreta o código como **texto puro**, e o exibe na página.
- Neste exemplo, o código em *JavaScript* substitui o conteúdo do elemento com **id=demo** pela data atual;
  - O método "**getElementById()**" acessa o elemento HTML cuja "**id**" é passada como parâmetro.
  - Note que o código em *JavaScript* foi colocado **depois** que o elemento "demo" foi criado.

O Objeto DOM

O objeto DOM

- Quando um navegador carrega um documento HTML, antes da exibição ele cria uma estrutura chamada *Document Object Model* (DOM) em memória:
  - Nesta estrutura estão representados todos os elementos do documento HTML (já combinados com os respectivos estilos definidos via CSS, se for o caso);
- Este DOM é acessível pelo programador via scripts através do objeto "**document**":
  - Provê acesso a ***todos*** os elementos HTML em uma página;
  - Tendo acesso a estes elementos, pode-se modificar seu conteúdo HTML, o valor de atributos específicos, o estilo (CSS), etc.

O objeto DOM

```
<html>
<body>

<h1>Primeiro exemplo de <i>Javascript</i></h1>

<p id="demo">Um parágrafo.</p>

<script type="text/javascript">
document.getElementById("demo").innerHTML=Date();
</script>

</body>
</html>
```

Primeiro exemplo de *Javascript*

Fri Mar 16 2012 15:11:30 GMT-0300 (E. South America Standard Time)

O objeto DOM

- Boa parte dos exemplos que serão apresentados ao longo deste tópico utiliza o objeto ***document*** para modificar o conteúdo e propriedades de elementos HTML;
- Para facilitar a programação em *JavaScript* empregando os métodos e propriedades de objetos, recomenda-se a utilização de uma IDE de programação;
- A especificação das propriedades e métodos disponíveis para o objeto ***Document*** pode ser encontrada em:
  - https://developer.mozilla.org/en-US/docs/Web/API/document
  - https://www.w3schools.com/jsref/dom_obj_document.asp

ONDE COLOCAR *SCRIPTS* EM UM DOCUMENTO?

Onde colocar *scripts* em um documento?

- *Scripts* podem ser colocados tanto no corpo (**<body>**) quanto no cabeçalho (**<head>**) de documentos HTML:
  - Não há limite para o número de *scripts* em um documento HTML;
  - Um mesmo documento pode ter *scripts* tanto no corpo quanto no cabeçalho, ao mesmo tempo;
  - É uma boa prática colocar todas as funções no cabeçalho ou no final do documento HTML → não interferem com o conteúdo da página.

Onde colocar *scripts* em um documento?

- *Scripts* também podem ser colocados em ***arquivos externos*** ao documento:
  - Análogo ao que ocorre com as folhas de estilo CSS;
  - O código presente em arquivos externos geralmente é utilizado em diferentes documentos HTML;
  - A extensão de arquivos externos *JavaScript* é "**.js**";
  - O arquivo externo **não** deve conter as *tags* **<script> </script>**;
  - É necessário que a *tag* <*script*> do cabeçalho do documento HTML "***aponte***" para o arquivo externo .js (atributo **src**):
    - **<script src=**"xxx.js"**></script>**

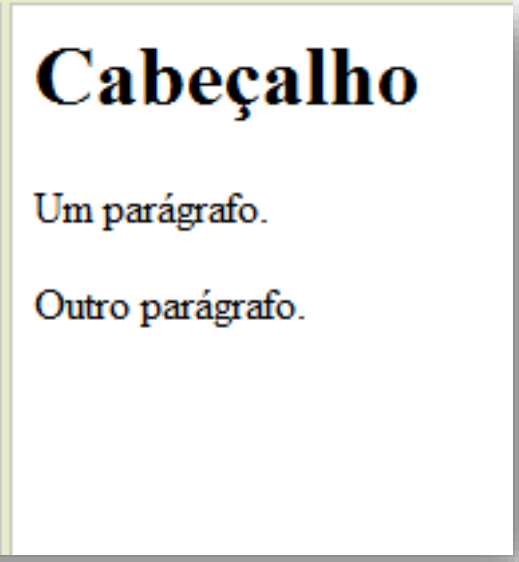
INSTRUÇÕES *JAVASCRIPT*

Instruções *JavaScript*

- Uma instrução *JavaScript* nada mais é que um comando dado ao navegador:
  - **Ex.**: document.write("Hello World!");
- O uso de **";"** ao final de uma instrução *JavaScript* **não** é obrigatório:
  - Mas permite que múltiplas instruções sejam colocadas em uma mesma linha;
- *JavaScript* é uma linguagem **case-sensitive**, ou seja, que **faz distinção** entre letras maiúsculas e minúsculas;
  - Requer cuidados com nomes de variáveis, objetos e funções;

Instruções *JavaScript*

```
<html>
<body>

<script type="text/javascript">
document.write("<h1>Cabeçalho</h1>");
document.write("<p>Um parágrafo.</p>");
document.write("<p>Outro parágrafo.</p>");
</script>

</body>
</html>
```



- Instruções em *JavaScript* podem estar envolvidas em "{" "}", formando um ***bloco de instruções***;
  - São usados para agrupar instruções pertencentes a funções ou condições (**if**s) e loops;

Instruções *JavaScript* – Variáveis

- Variáveis em *JavaScript* são declaradas com o comando "**var**" ou "**let**" (mais recente):
  - **Ex.**:
    - **var x**;
    - **let nome=**"Guilherme";
- Não é preciso definir tipos explicitamente;
- Os ***nomes*** de variáveis *JavaScript* **devem** começar com uma **letra**, **$** ou **_** (*underscore*);
- *JavaScript* é **case-sensitive** também em relação ao nome das variáveis;

Instruções *JavaScript* – Variáveis

- É possível atribuir ou não valores às variáveis em sua declaração:
  - **Ex.**:
    - **var x**; (não tem um valor associado → *undefined*)
    - **let nome**=”Guilherme”; (tem como valor a *string Guilherme*)
- *Strings* são definidas entre aspas (duplas ou simples);
- Valores booleanos são definidos com os valores “**true**” e “**false**” (sem as aspas);

Instruções *JavaScript* – Variáveis

- Variáveis declaradas ***dentro de funções → variáveis locais***:
  - Têm escopo local;
  - Deixam de existir quando se encerra a execução da função;
- Variáveis declaradas ***fora de funções → variáveis globais***:
  - Podem ser acessadas por quaisquer funções contidas no documento HTML;
  - Deixam de existir quando se fecha a página;
- JavaScript só possui ***escopo de bloco*** para variáveis criadas com "**let**".

Instruções *JavaScript* – Variáveis

- Quando se ***atribui um valor a uma variável não declarada previamente***:
  - Esta variável é criada automaticamente, como variável global;
- ***Constantes*** são declaradas com o comando "**const**" (ao invés de **var**);
  - **Não** podem ter o mesmo nome de variáveis ou funções que estão no mesmo escopo.

Instruções *JavaScript* – Variáveis

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Resultados aqui!</p>

<script>
// Nome das variaveis:
var firstName, lastName;

// Inicializacao:
firstName = "Guilherme";
lastName = "Coelho";
age = 15;

/* Resultado: */
document.getElementById("demo").innerHTML =
firstName + " " + lastName + " tem " + age + " anos.";
</script>

</body>
</html>
```

Guilherme Coelho tem 15 anos.

Instruções *JavaScript* – Operadores

- Operadores Aritméticos:
  - Supondo "**y=5**";

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

  - O operador "+" pode ser utilizado para concatenar *Strings*

Instruções *JavaScript* – Operadores

- Operadores de Comparação:
  - Supondo "**x=5**";

Operator	Description	Example
==	is equal to	x==8 is false x==5 is true
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

Instruções *JavaScript* – Operadores

- Operadores Lógicos:
  - Supondo "**x=6**" e "**y=3**";

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

- Operador Condicional:

```
variablename=(condition)?value1:value2
```

Instruções Condicionais

- Similares às de outras linguagens de programação:

```
if (condição)
{
    Código a ser executado caso a condição seja verdadeira;
}
else
{
    Código a ser executado caso a condição seja falsa;
}
```

Instruções Condicionais

- Similares às de outras linguagens de programação:

```
switch(n)
{
case 1:
  executa bloco 1
  break;
case 2:
  executa bloco 2
  break;
default:
  código a ser executado se n for diferente dos casos 1 e 2
}
```

Instruções de Repetição

- Similares às de outras linguagens de programação:

```
for (var=inicio; condição_de_parada; var=var+inc)
{
    Código a ser executado repetidamente;
}
```

```
while (condicao_de_repeticao)
{
    Código a ser executado repetidamente;
}
```

Instruções de Repetição

- Similares às de outras linguagens de programação:

```
do
{
    Código a ser executado repetidamente;
} while (condicao_de_repeticao)
```

- Instrução ***break***:
  - Encerra a execução de um *loop* e pula para a instrução imediatamente posterior ao loop (caso exista);
- Instrução ***continue***:
  - Encerra a execução corrente do *loop* e retoma a execução a partir da repetição seguinte.

Funções

- ***Funções*** são os blocos fundamentais de *JavaScript*;
  - São conjuntos de instruções que realizam uma tarefa ou calculam um valor;

```
function nome_da_funcao(var1, var2, ..., varX)
{
    Conjunto de instruções da função.
}
```

  - São executadas a partir de ***eventos*** ou ***chamadas***;

Funções

- Podem ser definidas tanto no **<head>** quanto no **<body>** de documentos HTML (além de documentos *externos* de scripts);
  - Recomenda-se sua definição no <**head**> do documento, para garantir que elas tenham sido carregadas pelos navegadores **antes** de serem chamadas.
- Funções podem, opcionalmente, retornar um valor, através da instrução ***return***;
- Quando os **parâmetros** são **valores primitivos**, eles são **passados por valor**, ou seja, modificações internas não se refletem fora da função;

Funções

- Quando os **parâmetros** são **objetos** e estes objetos têm **suas propriedades** modificadas dentro da função, tais modificações serão visíveis fora da função;
  - Caso um novo objeto seja atribuído ao parâmetro da função, isso **não** se reflete externamente → isto muda o ***valor*** do parâmetro.

Funções

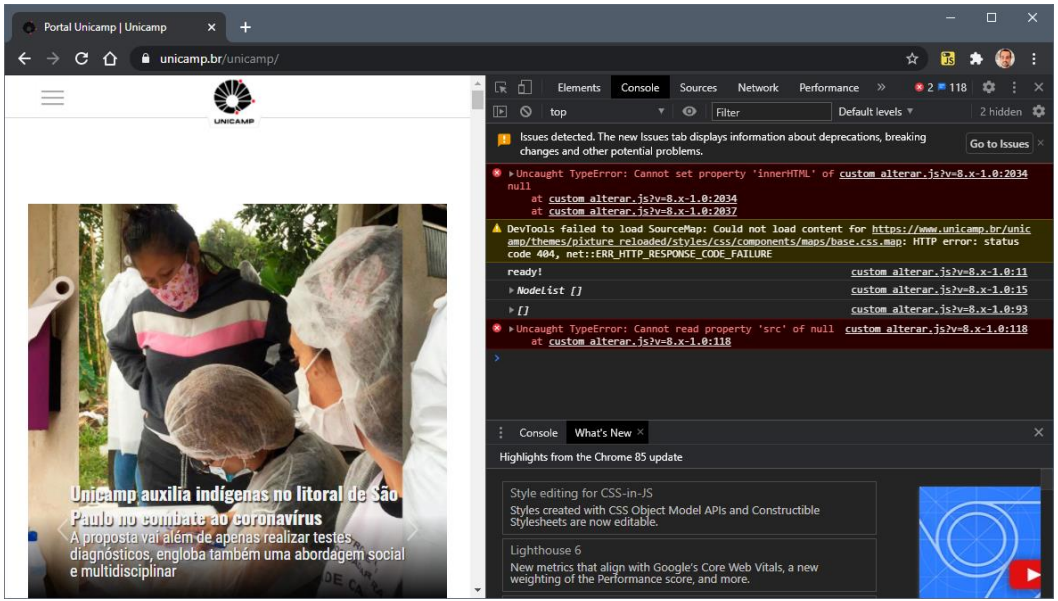
```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
a = 5;
return a*b;

}
</script>
</head>

<body>
<script type="text/javascript">
var g=4;
var h=3;
document.write(product(g,h));
document.write("<br />");
document.write(g);
</script>

</body>
</html>
```

```
15
4
```

Dica Final

Dica Final

- Quando estiver programando em JavaScript, visando a execução de seu código-fonte em um *browser*, **sempre** faça os testes com as ferramentas de desenvolvimento ativas;
  - Todos os erros e *warnings* serão exibidos na aba “Console”.



Referências

Referências

- [1] W3Schools JavaScript Tutorial:

  http://www.w3schools.com/js/default.asp

- [2] Standard ECMA-262: ECMAScript Language Specification

  http://www.ecma-international.org/publications/standards/Ecma-262.htm

- [3] Mozilla Developer Network – JavaScript:

  http://developer.mozilla.org/pt-BR/JavaScript