Aula 04

UWeb - Front-end

1. JavaScript

O que é?

Definição

- Linguagem de programação interpretada
- Principal linguagem *client-side*
- Utilizada para manipular HTML e CSS, a fim de programar o comportamento das páginas

Como Utilizar - Tag <script>

```
<script type="text/javascript">
    console.log("Hello World!");
</script>
```

Exemplo 1

Obs: console.log() é uma função que escreve no console do navegador.

Como Utilizar - Arquivo Externo

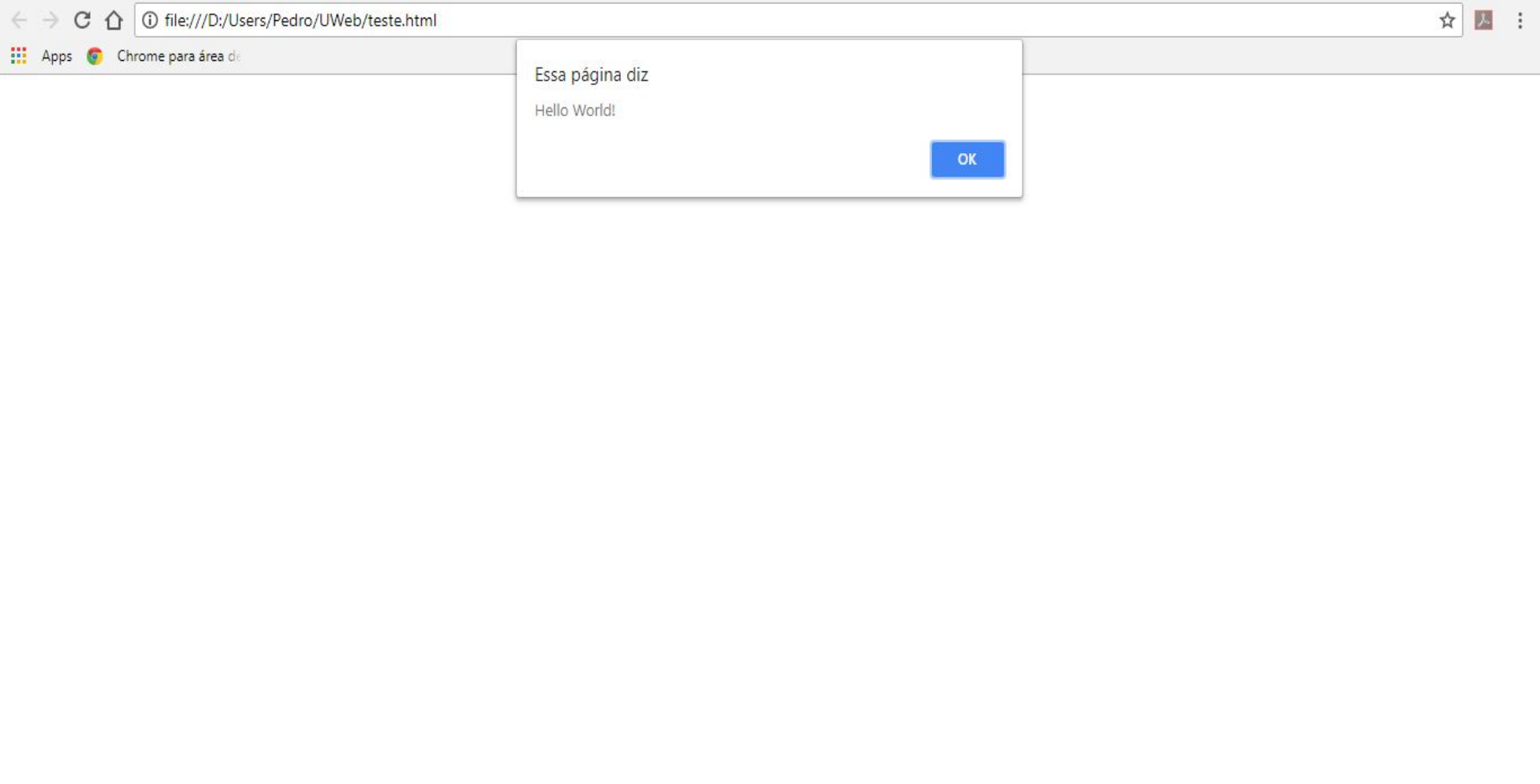
```
<script type="text/javascript" src=”script.js”></script>
```

Exemplo 2 - No HTML, especifica que o script é em JavaScript e o caminho para o arquivo

```
alert(“Hello World!”);
```

Exemplo 3 - No arquivo JavaScript

Obs: alert() é uma função que exibe um alerta no navegador com uma mensagem.



Alerta sendo exibido pelo navegador

Tipagem

```
var myVariable;
```

Em JavaScript, temos a presença da inferência de tipos e da tipagem dinâmica nas variáveis. Ou seja, se a variável myVariable estiver recebendo o valor de 1 em determinado ponto do script, o interpretador irá identificar que se trata de um tipo number(numérico, que pode ser inteiro ou decimal) e o programa irá funcionar normalmente. Também é possível trocar, ao longo do código, o valor que esta variável poderá receber, podendo por exemplo se tornar uma string (cadeia de caracteres), apenas recebendo uma string.

Tipos de dados

- Boolean é um tipo lógico, que pode assumir verdadeiro ou falso.
- Number representa um número inteiro ou decimal.
- String se trata de uma cadeia de caracteres.
- Array é uma estrutura que permite o armazenamento de vários valores em uma mesma referência.

FONTE: https://developer.mozilla.org/

Boolean

```
var x = true;
```

```
var y = false;
```

Number

```
var x1 = 14;          // Número inteiro
```

```
var x2 = 3.14159;   // Número decimal
```

```
var y = 123e5;        // Notação científica, 12300000
```

String

```
var primeiroNome = ‘Lucas’;       // Aspas simples
```

```
var segundoNome = ”Weiby”;   // Aspas duplas
```

```
var citacao = ”José uma vez disse, ‘bonito campo’. ”;
```

Array

```
var nomes = [“Weiby”, “Bruna”, “Freire”, “Camurça”];
```

```
var coisas = [true, 3.14159, “delta”];
```

Operadores

- **Aritméticos:**
  - **+** Adição
  - **-** Subtração
  - ***** Multiplicação
  - **/** Divisão
  - **%** Módulo (resto da divisão)
  - **++** Incremento (adiciona 1)
  - **--** Decremento (subtrai 1)

Operadores

- **Comparação:**
  - `==` igual a
  - `===` valor e tipo de dado iguais
  - `!=` diferente
  - `!==` valor e tipo de dados diferentes
  - `>` maior que
  - `<` menor que
  - `>=` maior que ou igual
  - `<=` menor que ou igual
  - `?` operador ternário

Operadores

- **Lógicos:**
  - **&&** “e” lógico (verdadeiro quando todas os operandos são verdadeiros)
  - **||** “ou” lógico (verdadeiro quando pelo menos um dos operandos é verdadeiro)
  - **!** “não” lógico (inverte o valor lógico do operando)

Estruturas condicionais

Muitas vezes, quando você escreve código, você quer executar ações diferentes para decisões diferentes. Você pode usar declarações condicionais em seu código para fazer isso. Em JavaScript temos as seguintes declarações condicionais:

- if , especifica o bloco de código a ser executado caso determinada condição seja atendida.
- else, acompanhado de um if, executa um bloco de código caso a condição seja falsa.
- else if, especifica uma nova condição a ser testada, caso a primeira seja falsa.

```
var ano = 2018;
if (ano == 2018) {
        console.log(“UWeb 2018 :)”);
}  else if (ano == 2017) {
        console.log(“Esse daí já passou”);
} else {
        console.log(“Sei nem em que ano eu tô”);
}
```

Loops

Loops podem executar um bloco de código um número de vezes.

- while repete um bloco de código enquanto uma condição especificada é verdadeira.
- for, semelhante ao while, porém em geral, o número de vezes que o loop vai repetir é especificado no se corpo.

While

```
var i = 0;
while (i < 5) {
        console.log("O número é " + i);
        i++;
}
```

For

```
for (i = 0; i < 5; i++) {
                console.log("O número é " + i);
}
```

Funções

- Uma função JavaScript é um bloco de código projetado para executar uma tarefa específica.
- Uma função permite o reuso de código.
- Uma função JavaScript é executada quando ela é invocada (chamada).
- Uma função JavaScript é definida com a palavra-chave function , seguida por um nome , seguido por parênteses ().

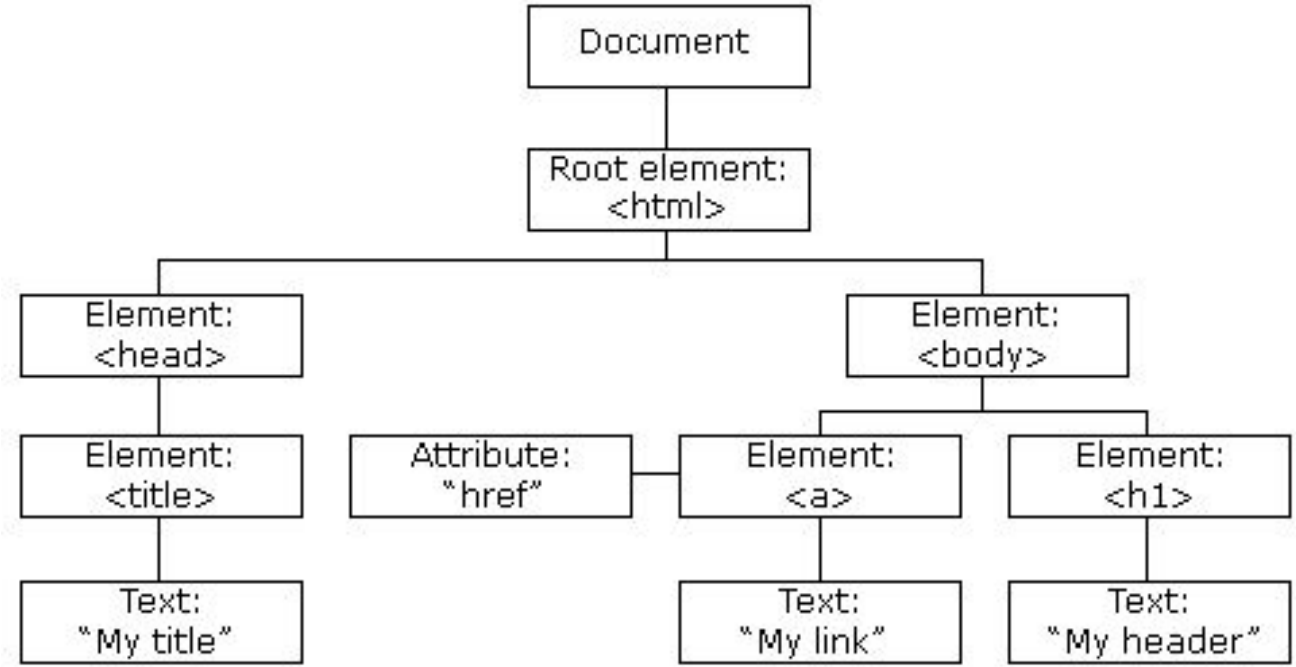
Funções

```
function funcao(a, b) {
    return a * b;  // Retorna o produto dos termos
}
var x = funcao(4, 3);  // Chamada da função
console.log(x);
```

DOM

- ***Document Object Model***
- Utilizado pelo navegador Web para representar a sua página Web
- Mais fácil trabalhar com DOM que diretamente com código HTML ou CSS
- Objeto ***document***: responsável por conceder ao código Javascript todo o acesso à árvore DOM do navegador Web

Árvore DOM


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

DOM

- **Documento**: nó mais importante do DOM
  - **Document:** usado por todos os documentos XML e outros que não sejam SVG (que também é um XML, porém com marcação já padronizada)
  - **HTMLDocument:** como o nome sugere, cuida de documentos HTML
  - **SVGDocument:** responsável pelos documentos SVG e também por outros documentos herdados da classe Document (como o Document.h e o HTMLDocument.h)
- **Elementos:** todas as tags que estão em arquivos HTML ou XML se transformam em elementos da árvore DOM. Um elemento é um nó com uma tag que pode ser usada para fazer subclasses específicas, as quais podem ser processadas de acordo com as necessidades da Render Tree (Element.h)
- **Texto:** É o texto que vai entre os elementos. Todo o conteúdo das tags

HTML DOM

- Modelo de objeto padrão e uma interface de programação para HTML
- Define:
  - Os elementos HTML como objetos
  - As propriedades de todos os elementos HTML
  - Os métodos para acessar todos os elementos HTML
  - Os eventos para todos os elementos HTML

- Padrão de como obter, alterar, adicionar ou excluir elementos HTML

Encontrando Elementos - ID

- Forma mais fácil de encontrar um elemento HTML no DOM

```
var myElement = document.getElementById("ID");
```

- Se o elemento for encontrado, o método retorna o elemento como um objeto (no myElement)
- Se o elemento não for encontrado, myElement conterá nulo.

Encontrando Elementos - Name

```
var myElement = document.getElementsByTagName("Name");
```

- Se o elemento for encontrado, o método retornará o elemento como um objeto (no myElement)
- Se o elemento não for encontrado, myElement conterá nulo.

Encontrando Elementos - Class

```
var myElement = document.getElementsByClassName("Class");
```

- Se o elemento for encontrado, o método retornará o elemento como um objeto (no myElement)
- Se o elemento não for encontrado, myElement conterá nulo.

Encontrando Elementos - Selector

```
var myElement = document.querySelectorAll("Seletor");
```

- Se o elemento for encontrado, o método retornará o elemento como um objeto (no myElement)
- Se o elemento não for encontrado, myElement conterá nulo.

Encontrando Coleções de Elementos

- **Exemplo:** Encontrar o elemento do formulário com id = "frm1":

```
var x = document.forms["frm1"];
```

- Se o elemento for encontrado, o método retornará o conjunto de elementos como um objeto (e será atribuído à variável x).
- Se o elemento não for encontrado, será atribuído nulo à variável x.

Eventos

- Um evento HTML pode ser algo que o navegador faz, ou algo que um usuário faz
- Aqui estão alguns exemplos de eventos HTML:
  - Uma página da Web HTML finalizou o carregamento
  - Um campo de entrada HTML foi alterado
  - Foi clicado um botão HTML
- Muitas vezes, quando os eventos acontecem, você pode querer fazer algo
- O JavaScript permite que você execute o código quando os eventos são detectados

Eventos HTML mais comuns

- **onchange:** Um elemento HTML mudou.
- **onclick**: O usuário clicou em um elemento HTML.
- **onmouseover**: O usuário move o mouse sobre um elemento HTML.
- **onmouseout**: O usuário move o mouse para fora de um elemento HTML.
- **onkeydown**: O usuário pressiona uma tecla do teclado.
- **onload**: O browser terminou de carregar a página interia.

FONTE:
https://www.w3schools.com/js/js_events.asp

Onload

- No arquivo .html

```
<body onload="checarCookies()">

<p id="demonstração"></p>
```

- No arquivo .js

```
function checarCookies() {
    var text = "";
    if (navigator.cookieEnabled == true) {
        text = "Cookies estão habilitados.";
    } else {
        text = "Cookies não estão habilitados.";
    }
    document.getElementById("demonstração").innerHTML = text;
```

Onchange

- No arquivo .html

```
Texto: <input type="text" id="fname" onchange="myFunction()">

<p>Ao clicar fora do input, uma função é chamada e transforma as
 letras do texto em maiúsculas.</p>
```

- No arquivo .js

```
function myFunction() {
    var x = document.getElementById("fname");
    x.value = x.value.toUpperCase();
}
```

Onclick

- No arquivo .html

```
<h1 id=”titulo” onclick="mostrarData(this)">Clique no texto!</h1>
```

- No arquivo .js

```
function displayDate() {
    document.getElementById("titulo").innerHTML = Date();
}
```

Onmousedown e Onmouseup

- No arquivo .js

```
function mDown(obj) {
    obj.style.backgroundColor = "#1ec5e5";
    obj.innerHTML = "Release Me";
}
function mUp(obj) {
    obj.style.backgroundColor="#D94A38";
    obj.innerHTML="Thank You";
}
```

Onmouseover e Onmouseout

- No arquivo .html

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)">Passe o
 mouse sobre mim</div>
```

- No arquivo .js

```
function mOver(obj) {
    obj.innerHTML = "Funciona!"
}
function mOut(obj) {
    obj.innerHTML = "Passe o mouse sobre mim"
}
```

Onmousedown e Onmouseup

- No arquivo .html

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)"

style="background-color:#D94A38;width:90px;height:20px;padding:
 40px;">

Clique aqui</div>
```

Exercício

- Em sua página de perfil, iniciada nos exercícios anteriores, crie uma seção abaixo da lista de hobbies, cujo conteúdo seja atualizado de acordo com o hobbie clicado.

Obrigadx!

Dúvidas?

petcomp@ufc.br