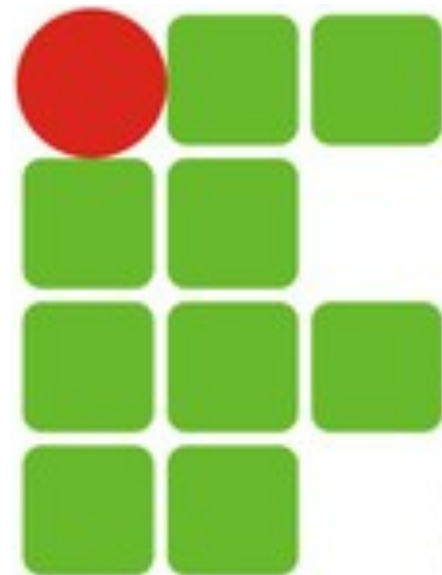


JAVASCRIPT

Prof. Fellipe Aleixo (fellipe.aleixo@ifrn.edu.br)



JAVASCRIPT

- É a linguagem de *script* da Web
- Adicionam funcionalidades às páginas HTML
 - Validação de entrada
 - Comunicação com servidores Web
 - Entre outros...
- Como qualquer linguagem de programação, você só irá aprender testando, faça isso com todos os exemplos
 - Teste também as suas variações
 - http://www.w3schools.com/js/tryit.asp?filename=tryjs_events

LINGUAGEM DE SCRIPT

- Trata-se de uma linguagem de programação “leve”
- O seu “código” pode ser inserido em páginas Web
- Esse código é executado pelos navegadores modernos
- É fácil de aprender

EXEMPLOS INICIAIS

- Código Javascript “escrevendo” na saída HTML

```
<script>
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph</p>");
</script>
```

- Javascript reagindo a eventos

```
<button type="button" onclick="alert('Welcome!')">Click Me!</button>
```

- Javascript mudando o código HTML

```
x = document.getElementById("demo") //Encontrar o elemento
x.innerHTML = "Hello JavaScript"; //Alterar o conteúdo
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_inner_html

EXEMPLOS INICIAIS

- Mudar dinamicamente a “origem” (src) de uma imagem

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_lightbulb

- Mudar o estilo de um elemento HTML (um atributo)

```
x = document.getElementById("demo") //Find the element
x.style.color = "#ff0000";          //Change the style
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_style

EXEMPLOS INICIAIS

- Verificar se a entrada fornecida pelo usuário é válida

```
if isNaN(x) {alert("Not Numeric")};
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_validate

JAVASCRIPT

- Código Javascript deve ser inserido entre as tags `<script>` e `</script>`
- Pode ser colocado no `<body>` ou `<head>` da página HTML

```
<script>  
alert("My First JavaScript");  
</script>
```

- Antigamente era necessário a definição de uma propriedade da tag `<script>` que era a `type="text/javascript"`
 - Não é mais necessário - Javascript é a linguagem de script padrão

FUNÇÕES E EVENTOS

- As instruções Javascript são executadas quando uma página é carregada
- Podemos desejar que um código seja executado quando um evento ocorrer, como o clique do mouse
- Se o código Javascript é colocado em uma função, esta função pode ser chamada quando um evento ocorrer

JAVASCRIPT NO <HEAD> OU <BODY>

- Você pode colocar vários código Javascript em um HTML
 - Podem ser colocados no <head> ou <body>, ou em ambos
- É uma prática comum que as funções Javascript sejam colocadas no cabeçalho, ou no final do corpo da página

JAVASCRIPT NO <HEAD> OU <BODY>

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML="1a. função JavaScript";
}
</script>
</head>

<body>
<h1>My Web Page</h1>

<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

JAVASCRIPTS EXTERNOS

- Scripts podem ser colocados em arquivos externos
- Arquivos externos podem conter códigos a serem utilizados por diversas páginas Web
- Arquivos externos Javascript possuem a extensão “.js”
- Ao utilizar scripts externos, a propriedade `src` da tag `<script>` deve apontar para esse arquivo

JAVASCRIPTS EXTERNOS

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

<h1>My Web Page</h1>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> myFunction is stored in an external file
called "myScript.js".</p>

<script src="myScript.js"></script>

</body>
</html>
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_externalexample

MANIPULANDO ELEMENTOS HTML

- Javascript é utilizada tipicamente para a manipulação de elementos HTML
- Para acessar elementos HTML, é utilizado o método `document.getElementById(id)`
- O atributo “id” é utilizado para identificar o elemento HTML

MANIPULANDO ELEMENTOS HTML

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

<h1>My First Web Page</h1>

<p id="demo">My First Paragraph</p>

<script>
document.getElementById("demo").innerHTML="My First JavaScript";
</script>

</body>
</html>
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom

ESCREVENDO DA SAÍDA HTML

- Exemplo - escrever um elemento `<p>` na saída HTML

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

<h1>My First Web Page</h1>

<script>
document.write("<p>My First JavaScript</p>");
</script>

</body>
</html>
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_write

WARNING

- O método `document.write()` é utilizado para escrever diretamente na saída do documento
- Se esse método for executado após o documento ser carregado, a página HTML em questão será sobrescrita

WARNING

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

<h1>My First Web Page</h1>
<p>My First Paragraph.</p>
<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    document.write("Oops! The document disappeared!");
}
</script>

</body>
</html>
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_write_over

INSTRUÇÕES JAVASCRIPT

- Instruções Javascript são comandos para o navegador
- O propósito das instruções é dizer ao navegador o que fazer

```
document.getElementById("demo").innerHTML="Hello Dolly";
```

- Essa instrução solicita ao navegador escrever “Hello Dolly” dentro do elemento HTML identificado com o id=“demo”
- O ponto e vírgula é usado para separar as instruções
 - Com “;” mais de uma instrução pode ser adicionada em uma linha

BLOCOS DE CÓDIGO

- Instruções Javascript pode ser agrupadas em blocos de código
- Blocos iniciam com “abre chave” - “{“ e finalizam com “fecha chave” - “}”
- O propósito de um bloco é fazer com que uma sequência de instruções executem juntas
 - Um bom exemplo são as funções

RESTRICÇÕES JAVASCRIPT

- Javascript diferencia maiúsculas das minúsculas
 - A função `getElementById` ≠ `getElementbyID`
 - A variável `myVariable` ≠ `MyVariable`
- Javascript ignora espaços em branco “extras”
- É possível a quebra de uma instrução que está utilizando uma “*string*” de texto – dividindo esta *string*

```
document.write("Hello \  
World!");
```

COMENTÁRIOS JAVASCRIPT

- Comentários de uma única linha iniciam com “//”

```
// Write to a heading:  
document.getElementById("myH1").innerHTML="Welcome to my Homepage";  
// Write to a paragraph:  
document.getElementById("myP").innerHTML="First paragraph";
```

- Comentários de várias linhas iniciam com “/*” e finalizam com “*/”

```
/*  
The code below will write  
to a heading and to a paragraph,  
and will represent the start of  
my homepage:  
*/  
document.getElementById("myH1").innerHTML="Welcome to my Homepage";  
document.getElementById("myP").innerHTML="First paragraph";
```

VARIÁVEIS JAVASCRIPT

- Variáveis podem ser usadas para armazenar valores ($x = 5$) ou expressões ($z = x + y$)
- As variáveis podem possuir nomes curtos, e seus nomes
 - Devem ser iniciar com uma letra
 - Nomes de variáveis também podem iniciar com “\$” e “_”
 - São diferenciadas as maiúsculas das minúsculas
- Tipos da dados
 - Tipo texto (*string*) – valor entre aspas duplas (ou simples)
 - Tipo numérico
 - Outros: Boolean, Array, Object, Null, Undefined

VARIÁVEIS JAVASCRIPT

- Exemplos:

```
var pi = 3.14;  
var person = "John Doe";  
var answer = 'Yes I am!';
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_data2

DECLARANDO VARIÁVEIS JAVASCRIPT

- Através da utilização da palavra-chave “var”
 - Ex.: `var carro;`
- Após a declaração a variável está vazia (`undefined`)
- Para atribuir valores utiliza-se o operador “=”
 - `carro = "Chevrolet";`
- Também é possível declarar e atribuir valor inicial
 - `var carro = "Chevrolet";`

DECLARANDO VARIÁVEIS JAVASCRIPT

- Exemplo:

```
<p id="demo"></p>  
var carname="Volvo";  
document.getElementById("demo").innerHTML = carname;
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_variables1

- A instrução var pode ser usada para declarar várias variáveis

```
var lastname = "Doe", age = 30, job = "carpenter";  
// ou  
var lastname = "Doe",  
age = 30,  
job = "carpenter";
```

DECLARANDO VARIÁVEIS JAVASCRIPT

```
var answer = "It's alright";
var answer = "He is called 'Johnny'";
var answer = 'He is called "Johnny"';

var x1 = 34.00;      // Written with decimals
var x2 = 34;        // Written without decimals

var y = 123e5;      // 12300000
var z = 123e-5;    // 0.00123

var x = true;
var y = false;

var cars = new Array();
cars[0] = "Saab";
cars[1] = "Volvo";
cars[2] = "BMW";
// ou
var cars = new Array("Saab", "Volvo", "BMW");
```

TIPOS DINÂMICOS

- As variáveis Javascript possuem tipos dinâmicos
 - Uma mesma variável pode armazenar dados de diferentes tipos

```
var x;           // Now x is undefined
var x = 5;      // Now x is a Number
var x = "John"; // Now x is a String
```

OBJETOS JAVASCRIPT

- Um objeto é definido entre abre e fecha chaves
- As propriedades de um objeto são definidas em pares “nome” e “valor”, separados por vírgula

```
var person={firstname:"John", lastname:"Doe", id:5566};
```

```
var person={  
  firstname : "John",  
  lastname  : "Doe",  
  id        : 5566  
};
```

```
// Acesso às propriedades de um objeto  
name = person.lastname;  
name = person["lastname"];
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_object

UNDEFINED E NULL

- `undefined` é o valor de uma variável sem valor atribuído
- Uma variável pode ser esvaziada, atribuindo-se `null`

- Ex.:

```
cars = null;  
person = null;
```

- Declarando tipos de variáveis - utilizando a expressão `new`

```
var carname    = new String;  
var x          = new Number;  
var y          = new Boolean;  
var cars       = new Array;  
var person     = new Object;
```

PROPRIEDADES E MÉTODOS

- Propriedades são “valores” associados à objetos
- Métodos são “ações” que um objeto pode realizar
- Exemplo: um objeto “carro”
 - Pode incluir as propriedades: nome, modelo, peso, cor, etc.
 - Pode incluir os métodos: ligar(), acelerar(), frear(), etc.
- Quase tudo em Javascript são objetos
 - Strings, Functions, Arrays, Dates...

PROPRIEDADES E MÉTODOS

- Objeto *String*:

```
var txt = new String("Hello World");
```

- Exemplo de propriedade - `txt.length`
- Exemplo de método - `txt.indexOf("World")`

CRIANDO OBJETOS

- É possível criar objetos e ir adicionando propriedades ao mesmo

```
person = new Object();  
person.firstname = "John";  
person.lastname = "Doe";  
person.age = 50;  
person.eyecolor = "blue";
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_create_object

CRIANDO OBJETOS

- Para acessar propriedades de objetos
 - `<nome do objeto>.<nome da propriedade>`
- Da mesma forma para os métodos
- Exemplos:

```
var message = "Hello World!";  
var x = message.length;  
  
var message = "Hello world!";  
var x = message.toUpperCase();
```

FUNÇÕES JAVASCRIPT

- Bloco de código que é executado quando “alguém” chama
 - Uma função pode ser invocada em resposta a um evento

■ Sintaxe:

```
function functionName(argument1, argument2) {  
    some code to be executed;  
}
```

- Exemplo (http://www.w3schools.com/js/tryit.asp?filename=tryjs_function3):

```
<button onclick="myFunction('Harry Potter','Wizard')">Try #1</button>  
<button onclick="myFunction('Bob','Builder')">Try #2</button>  
<script>  
function myFunction(name,job) {  
    alert("Welcome " + name + ", the " + job);  
}  
</script>
```

FUNÇÕES JAVASCRIPT

- Funções podem ter um valor de retorno
 - Instrução `return`

```
function myFunction() {  
    var x = 5;  
    return x;  
}  
...  
var myVar = myFunction();
```

- Exemplo (http://www.w3schools.com/js/tryit.asp?filename=tryjs_function_return):

```
function myFunction(a,b) {  
    return a * b;  
}  
  
document.getElementById("demo").innerHTML=myFunction(4,3);
```

FUNÇÕES JAVASCRIPT

- O `return` também pode ser utilizado para antecipar o final de uma função – o retorno de um valor é opcional
- Variáveis globais vs. Variáveis locais
 - Variável declarada em uma função se torna local – só podendo ser acessada dentro da mesma
 - Variáveis de mesmo nome podem ser usadas em diferentes funções
 - As variáveis locais são deletadas ao término da execução da função
 - Variáveis declaradas fora das função são globais – acessíveis a partir de todo *script* de uma página, bem como suas funções
- Ao atribuir valor a uma variável não declarada (`var`) a mesma é automaticamente declarada como global

OPERADORES JAVASCRIPT

- Atribuição → “=”
 - $X += Y \rightarrow X = X + Y$
 - $X -= Y \rightarrow X = X - Y$
 - $X *= Y \rightarrow X = X * Y$
 - $X /= Y \rightarrow X = X / Y$
 - $X \% = Y \rightarrow X = X \% Y$
- Aritméticos
 - Adição → “+”
 - Subtração → “-”
 - Multiplicação → “*”
 - Divisão → “/”
 - Resto da divisão inteira (módulo) → “%”
 - Incremento → “++”
 - Decremento → “--”
- Concatenação de strings → “+”

OPERADORES JAVASCRIPT

- Adicionando *strings* e números – quais os resultados?

```
x = 5 + 5;  
y = "5" + 5;  
z = "Hello" + 5;
```

- Confira: http://www.w3schools.com/js/tryit.asp?filename=tryjs_operators5

OPERADORES JAVASCRIPT

■ Comparações no Javascript (dado x = 5):

Operador	Descrição	Exemplo	Resultado
==	Igual a	x == 8	false
		x == 5	true
===	Exatamente igual a (tipo e valor)	x === "5"	false
		x === 5	true
!=	Diferente	x != 8	true
!==	Diferente no tipo ou valor	x !== "5"	true
		x !== 5	false
>	Maior que	x > 8	false
<	Menor que	x < 8	true
>=	Maior ou igual	x >= 8	false
<=	Menor ou igual	x <= 8	true

OPERADORES JAVASCRIPT

- Operadores lógicos (dados $x = 6$ e $y = 3$):

Operador	Descrição	Exemplo
&&	“and” lógico	$x < 10 \ \&\& \ y > 1$) is true
	“or” lógico	$(x==5 \ \ y==5)$ is false
!	“not” lógico	$!(x==y)$ is true

- Operador condicional

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

```
voteable = (age<18)?"Too young":"Old enough";
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison

CONDICIONAIS

■ Instrução **if**

```
if (time < 20) {  
    x = "Good day";  
}
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_ifthen

■ Instrução **if...else**

```
if (time < 20) {  
    x = "Good day";  
} else {  
    x = "Good evening";  
}
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_ifthenelse

CONDICIONAIS

- Instrução **if...else if...else**

```
if (time < 10) {  
    x = "Good morning";  
} else if (time < 20) {  
    x = "Good day";  
} else {  
    x = "Good evening";  
}
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_elseif

CONDICIONAIS

■ Instrução **switch**

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_switch

- A palavra-chave `default` pode ser utilizada para definir as operações a serem realizadas quando nenhuma das opções for satisfeita

```
var day = new Date().getDay();
switch (day) {
  case 0:
    x="Today it's Sunday";
    break;
  case 1:
    x="Today it's Monday";
    break;
  case 2:
    x="Today it's Tuesday";
    break;
  case 3:
    x="Today it's Wednesday";
    break;
  case 4:
    x="Today it's Thursday";
    break;
  case 5:
    x="Today it's Friday";
    break;
  case 6:
    x="Today it's Saturday";
    break;
}
```

LAÇO FOR

- Repete um dado bloco de instruções uma determinada quantidade de vezes

```
for (var i = 0; i < cars.length; i++) {  
    document.write(cars[i] + "<br>");  
}
```

```
for (var i = 0, len = cars.length; i < len; i++) {  
    document.write(cars[i] + "<br>");  
}
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for

LAÇO FOR/IN

- Itera nas propriedades de um objeto

```
var person={ fname:"John",lname:"Doe",age:25};  
  
for (x in person) {  
    txt = txt + person[x];  
}
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_object_for_in

- Itera segundo uma determinada condição

```
while (i < 5) {  
    x = x + "The number is " + i + "<br>";  
    i++;  
}
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_while

LAÇO DO-WILE

- Semelhante ao `while`, só que com o teste no final do laço

```
do {  
    x = x + "The number is " + i + "<br>";  
    i++;  
}  
while (i < 5);
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_dowhile

BREAK E CONTINUE

- Instruções `break` e `continue` possibilitam sair de um laço

- Break

- Utilizada no `switch`, após a execução de uma das opções
- O laço é interrompido, e a execução continua após o mesmo

```
for (i = 0; i < 10; i++) {  
    if (i == 3) {  
        break;  
    }  
    x = x + "The number is " + i + "<br>";  
}
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_break

BREAK E CONTINUE

■ Continue

- Interrompe a execução da iteração corrente de um laço
- E continua com a “próxima” iteração do laço

```
for (i = 0; i <= 10; i++) {  
    if (i == 3) continue;  
    x = x + "The number is " + i + "<br>";  
}
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_continue

ERROS

- Vários erros podem ocorrer na execução de um Javascript
 - Exemplos:
 - Erros de sintaxe
 - Entradas erradas
 - Falta de suporte a alguma das características da linguagem
- Quando algo dá errado, o “motor” Javascript para e gera uma mensagem de erro
 - Lança (*throw*) um erro

TRY-CATCH

- A instrução `try` permite a definição de um bloco de código que espera (de alguma forma) que erros aconteçam
- A instrução `catch` permite a definição de um bloco de código que será executado caso ocorra um erro no bloco `try`
- **Sintaxe:**

```
try {  
    //Run some code here  
} catch(err) {  
    //Handle errors here  
}
```

TRY-CATCH

```
<!DOCTYPE html>
<html>
<head>
<script>
var txt="";
function message()
{
  try {
    addlert("Welcome guest!");
  } catch(err) {
    txt = "There was an error on this page.\n\n";
    txt += "Error description: " + err.message + "\n\n";
    txt += "Click OK to continue.\n\n";
    alert(txt);
  }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()">
</body>
</html>
```

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_try_catch

THROW

- A instrução throw permite lançar erros customizados
 - Exemplo (http://www.w3schools.com/js/tryit.asp?filename=tryjs_throw_error):

```
<script>
function myFunction() {
  try {
    var x = document.getElementById("demo").value;
    if(x == "") throw "empty";
    if(isNaN(x)) throw "not a number";
    if(x > 10) throw "too high";
    if(x < 5) throw "too low";
  } catch(err) {
    var y = document.getElementById("mess");
    y.innerHTML = "Error: " + err + ".";
  }
}
</script>

<h1>My First JavaScript</h1>
<p>Please input a number between 5 and 10:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="mess"></p>
```

VALIDAÇÃO

- Javascript pode ser utilizado para validar a entrada do usuário antes que esta seja enviada para o servidor
 - Campos obrigatórios foram deixados em branco?
 - O usuário informou um e-mail válido?
 - O usuário informou uma data válida?
 - O usuário entrou com um texto em um campo numérico?
- Campos requeridos (http://www.w3schools.com/js/tryit.asp?filename=tryjs_form_validation):

```
function validateForm() {  
    var x = document.forms["myForm"]["fname"].value;  
    if (x == null || x == "") {  
        alert("First name must be filled out");  
        return false;  
    }  
}
```

VALIDAÇÃO

- Validação de e-mail:

- http://www.w3schools.com/js/tryit.asp?filename=tryjs_form_validate_email

```
function validateForm() {
    var x = document.forms["myForm"]["email"].value;
    var atpos = x.indexOf("@");
    var dotpos = x.lastIndexOf(".");
    if (atpos < 1 || dotpos < atpos+2 || dotpos+2 >= x.length) {
        alert("Not a valid e-mail address");
        return false;
    }
}
```

REFERÊNCIAS JAVASCRIPT

- Você pode encontrar referências sobre os “objetos” javascript, objetos ligados ao navegador e objetos HTML DOM
 - Estas referências contêm exemplos de cada objeto, suas propriedades e seus métodos
 - Objetos internos do Javascript
 - Objetos ligados ao navegador
 - Objetos HTML DOM
 - <http://www.w3schools.com/jsref/default.asp>

JAVASCRIPT

- Aprenda Javascript através de exemplos
 - JavaScript Examples
 - http://www.w3schools.com/js/js_examples.asp
 - JavaScript Objects Examples
 - http://www.w3schools.com/js/js_ex_objects.asp
 - JavaScript Browser Objects Examples
 - http://www.w3schools.com/js/js_ex_browser.asp
 - JavaScript HTML DOM Examples
 - http://www.w3schools.com/js/js_ex_dom.asp
- Depois teste os seus conhecimentos com um **quiz**
 - <http://www.w3schools.com/quiztest/quiztest.asp?qtest=JavaScript>