MUNDO

JavaScript™

Seja Bem Vindo ao mundo JAVASCRIPT!

O objetivo desse **ESTUDO DIRIGIDO** é fazermos um 1º contato com a linguagem javascript e sua sintaxe. E despertar a curiosidade e interesse de vocês no aprendizado de nosso próximo conteúdo.

Nesse momento, não preocupe-se caso surjam dúvidas ou dificuldades de entendimento em algum assunto apresentado em algum slide desse estudo dirigido.

Iremos, passo a passo, exercitar esses conteúdos na prática durante as aulas.

JAVASCRIPT é uma das linguagens mais interessantes e solicitadas pelo mercado de trabalho para desenvolvimento de aplicações Web.
Temos muito o que aprender e praticar!
Hoje estamos só começando…
Boa leitura!

Iniciando…

Primeiramente, é importante dizer que o JAVASCRIPT não é JAVA!

Trata-se de uma linguagem de programação baseada em scripts utilizada no desenvolvimento de aplicações Web.

É uma linguagem interpretada, ou seja, o código-fonte é lido e executado diretamente pelo navegador no momento que é carregado através de uma página web.

Por ser uma linguagem interpretada não é gerado um arquivo executável.

JavaScript consegue interagir com todos os elementos de uma página HTML.

Consegue trabalhar com variáveis e modificar a aparência de elementos/objetos sem a necessidade de ficar recarregando a página.

É interessante saber também que o *JavaScript* é uma linguagem orientada a eventos.

Como assim?

Eventos são ações que ocorrem na página web, sejam pela interação do usuário ou eventos gerados pelo sistema (navegador, sistema operacional, etc.).

Exemplos de eventos:

* Clique num botão da página web;
* Preenchimento de um campo de formulário;
* Carregar uma página web no navegador.

Portanto, conseguiremos programar o comportamento de um botão, de um campo em um formulário fazendo chamada de funções.

Assim como temos funções em PYTHON também podemos trabalhar com funções em JAVASCRIPT.

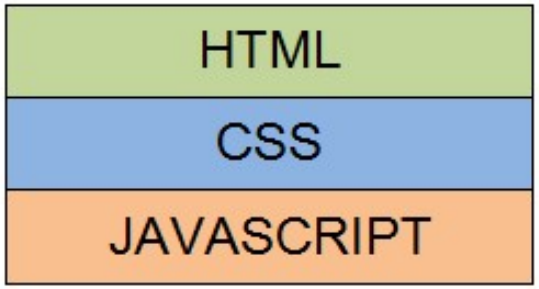
FUNCIONALIDADES GERAIS DO JAVASCRIPT

- **INTERAÇÃO COM FORMULÁRIOS**: com o Javascript é possível acessar campos e valores digitados num formulário HTML e fazer a validação dos mesmos realizando cálculos e sugestões de preenchimento dos campos de dados;
- **INTERAÇÃO COM LINGUAGENS DINÂMICAS**: podemos usar Javascript com outras linguagens de programação Web.
- **CONFORMIDADE COM PADRÕES WEB**: dois conceitos surgiram com o conceito de desenvolvimento com conformidade com os padrões Web:

  - ✓ Javascript não obstrutivo
  - ✓ Melhoria progressiva

JAVASCRIPT e os padrões WEB

É a terceira camada do bolo das tecnologias padrões da web, duas das quais (HTML e CSS). Dentro de uma mesma página Web podemos encontrar HTML, CSS e código JavaScript.



HTML	É a linguagem de marcação usada para estruturar o conteúdo de uma página Web tais como parágrafos, cabeçalhos, tabelas de conteúdo, imagens e vídeos.
CSS	É a linguagem de regras de estilo usada para desenvolver o design de uma página HTML. No CSS conseguimos definir cores de fundo e fontes, tipos e tamanhos de fontes, hierarquia de menus, posicionamentos de elementos de páginas tais como imagens, texto e tabelas.
JAVASCRIPT	É a linguagem de programação que possibilita a criação de conteúdo que atualiza dinamicamente, controla multimídias, imagens animadas, arquivos, menus, etc.

MODULARIZAÇÃO e ORGANIZAÇÃO

Conseguimos trabalhar com modularização de código em JAVASCRIPT.

O que é modularização de código?

Modularização ou componentização é o mecanismo que permite um sistema ser dividido em partes que interagem entre sí.
Chamamos essas partes de módulos.

A modularização é interessante porque possibilita reutilização da mesma em outros sistemas facilitando a manutenção e reuso de código.

MODULARIZAÇÃO e ORGANIZAÇÃO

Os benefícios da modularização são:

- Dividir problemas grandes em vários problemas menores, de baixa complexidade.
    - Número pequeno de variáveis
    - Poucos caminhos de controle (caminhos do início ao fim)
- Possibilidade de utilizar soluções gerais para classes de problemas em lugar de soluções específicas para problemas particulares.
    - Reusabilidade
    - Solucionar uma única vez o problema
- Permite delimitar o escopo (nível de abrangência) de variáveis.
    - Variáveis locais
- Evita a repetição, dentro de um mesmo algoritmo, de uma seqüência de ações em diferentes pontos.

MODULARIZAÇÃO e ORGANIZAÇÃO

Para fazer bem uma modularização de código é necessário:

1) Conhecer a linguagem que será utilizada;
2) Conhecer exatamente o que deseja-se implementar/codificar – objetivo da aplicação;
3) Definir uma organização da aplicação. Pensar em blocos de código (em componentes);
4) Pensar como esses componentes irão interagir entre sí na aplicação;
5) Pensar em modularidade implicar em não referenciar objetos/estados globais e sim passar como argumentos da função construtora (facilita os testes);
6) Cada bloco/módulo poderá estar num arquivo distinto, separado da aplicação (arquivos com poucas linhas);
7) Colocar comentários no seu código!
8) Planejar/organizar a integração dos módulos implementados;
9) Elaborar a gama de testes manuais ou automatizados dos módulos e da aplicação completa.

SINTAXE JAVASCRIPT

SINTAXE JAVASCRIPT

Nesse momento iremos nos familiarizar um pouco com a sintaxe JAVASCRIPT.

Os objetos da linguagem Javascript podem ser agrupados em três categorias:

- **Objetos internos**: strings, arrays, datas, etc;
- **Objetos de ambiente**: objeto window e document;
- **Objetos personalizados**: desenvolvidos pelo desenvolvedor.

Sabe o que é um **OBJETO** em programação?

Caso não saiba, fique tranquilo!
Falaremos sobre isso nas atividades de aula.

Por hora, pense que um objeto em programação é uma função com **características** e **comportamentos**.

Por exemplo: função CALCULO_MATEMÁTICO.
Essa função tem a característica de precisar receber um valor do tipo inteiro.
E ela tem o comportamento de calcular a raíz quadrada do número recebido.
Depois de calcular essa função devolve o resultado do cálculo que no caso é o valor da raíz quadrada do número recebido.

USANDO EVENTOS DO JAVASCRIPT

Eventos são muito usados em Javascript e ajudam a criar a interatividade em uma aplicação Web. Exemplos de eventos "nativos" do Javascript:

Evento	É disparado...
click	quando é pressionado e liberado o botão primário do mouse, trackpad, etc.
mousemove	sempre que o cursor do mouse se move.
mouseover	quando o cursor do mouse é movido para sobre algum elemento.
mouseout	quando o cursor do mouse se move para fora dos limites de um elemento.
dblclick	quando acontece um clique duplo com o mouse, trackpad, etc.
DOMContentLoaded	quando o DOM do documento está totalmente carregado (mais sobre isso num tutorial futuro).
load	quando todo o documento (DOM e recursos externos como imagens, scripts, etc) está totalmente carregado.
keydown	quando uma tecla no teclado é pressionada.
keyup	quando uma tecla no teclado é liberada (depois de pressionada).
scroll	quando há "rolagem" (scroll) num elemento.

PALAVRAS CHAVES DO JAVASCRIPT

- break
- case
- catch
- class
- const
- continue
- debugger
- default
- delete
- do
- else
- export
- extends
- finally
- for
- function
- if
- import
- in
- instanceof
- new
- return
- super
- switch
- this
- throw
- try
- typeof
- var
- void
- while
- with
- yield

PALAVRAS RESERVADAS DO JAVASCRIPT

- `abstract`
- `boolean`
- `byte`
- `char`
- `double`
- `final`
- `float`
- `goto`
- `int`
- `long`
- `native`
- `short`
- `synchronized`
- `throws`
- `transient`
- `volatile`

OPERADORES EM JAVASCRIPT

Operador	*Explicaçao*
+ - * / %	Soma, subtração, multiplicação, divisão, resto da divisão
+	Concatenação de Strings
< > >= <=	Maior, menor, maior ou igual, menor ou igual
++ --	Incrementar, decrementar
== !=	Igualdade, desigualdade
=== !==	Identidade, não identidade
&& ||	AND, OR
?:	Retorna true ou false em uma expressão condicional
=	Atribuição de conteúdo a uma variável
.	Acessar propriedade ou método de um objeto
()	Chamando uma função
[]	Indexação de vetores (array)
instanceof	Retorna o tipo do objeto
new	Criação de um objeto
typeof	Retorna o tipo de dado de uma variável
void	Retorna um valor indefinido de uma função

PRECEDÊNCIA DOS OPERADORES EM JAVASCRIPT

Precedëncia	*Operador*
1º	. [] new
2º	()
3º	++ --
4º	* / %
5º	+ -
6º	< > >= <= instanceof
7º	== != === !==
8º	&&
9º	||
10º	?:

Operações matemáticas:

Operador	*Explicação*
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão
++	Increment
--	Decrement

OPERADORES DE COMPARAÇÃO EM JAVASCRIPT

Operador	*Explicação*	*Exemplo*
==	Igualdade	var x = 27 x == 27 (retorna true) x == 21 (retorna false)
===	Identidade	var x = 27 x === 27 (retorna true) x === 21 (retorna false)
!=	Desigualdade	var x = 30 x != 27 (retorna true) x != 30 (retorna false)
!==	Nao identidade	var x = 30 x !== 27 (retorna true) x !== “30” (retorna false)
>	Maior que	var x = 30 x > 20 (retorna true) x > 100 (retorna false)
<	Menor que	var x = 30 x < 20 (retorna false) x < 100 (retorna true)
>=	Maior que ou igual a	var x = 30 x >= 20 (retorna true) x >= 100 (retorna false)
<=	Menor que ou igual a	var x = 30 x <= 20 (retorna false) x <= 100 (retorna true)

OPERADORES LÓGICOS EM JAVASCRIPT

Operadores lógicos:

Operador	*Explicação*	*Exemplo*
&&	AND	var x = 27 var y = 10 (x > y) && (y < 10) (retorna false)
||	OR	var x = 27 var y = 10 (x > y) || (y < 10) (retorna true)
!	NOT	var x = 27 ! (x > 30) (retorna true)

OPERADORES LÓGICOS EM JAVASCRIPT

Variável A	*Operador lógico*	*Variavel B*	*Resultado*
(A > 5)	AND	(B <= 100)	TRUE
(A > 5)	AND	(B < 100)	FALSE
(A < 5)	AND	(B > 10)	FALSE
(A < 5)	AND	(B < 100)	FALSE
(A > 5)	OR	(B <= 100)	TRUE
(A > 5)	OR	(B < 100)	TRUE
(A < 5)	OR	(B > 10)	TRUE
(A < 5)	OR	(B < 100)	FALSE
	NOT	(B <= 100)	FALSE
	NOT	(A < 5)	TRUE

OPERADORES DE ATRIBUIÇÃO EM JAVASCRIPT

Operação	*Explicação*
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /+ y	x = x / y
x <<= y	x = x << y
x &= y	x = x & y

E aí, achou alguma similaridade com PYTHON?

Nos slides anteriores vimos detalhes da sintaxe da linguagem JAVASCRIPT que encontramos em outras linguagens de programação também.

Ao longo do seu processo de aprendizado em programação verás que toda linguagem tem os mesmos recursos.

O que muda é a sintaxe, ou seja, a forma de escrever essa linguagem de programação.

Quer um exemplo?
Lá vai...
Veja os blocos de código abaixo:

Código em PYTHON

```python
c = 1
while c <= 10:
    print(c)
    c += 1
```

Código em JAVASCRIPT

```javascript
<script>
  let c = 1;
  while (c <= 10) {
    document.write(c + "<br>");
    c++;
  }
</script>
```

Os dois códigos produzem o mesmo resultado? SIM.

A questão aqui é a sintaxe. Esses códigos estão em linguagens de programações diferentes. Logo, as sintaxes são diferentes!

Vamos agora entender o que é um PONTO DE INTERROGAÇÃO em JAVASCRIPT....

USO DO CONDICIONAL “?” EM JAVASCRIPT

```
<script>
  var meuNome;
   var liberacaoAcesso = "Acesso liberado, Sr/Sra " +
          (meuNome ? meuNome : "usuário desconhecido!");
   alert(liberacaoAcesso);
</script>
```

A sintaxe do operador “?:” é valor = (expressão) ? operando1 : operando2;
Se a expressão for TRUE, o retorno será o operando.
Caso contrário, será o operando2.

ESTRUTURAS DE CONTROLE EM JAVASCRIPT

Declaração	*Finalidade*
function	declaração de uma função
return	Retorno o valor de uma função
if... else	Criação de comando condicional
switch	Criação de comando condicional
case	Usando comando condicional switch
break	Usando no switch para validação de valores
default	Usando no switch para valores padrões
for	Criação de comando de repetição
continue	Reiniciar um comando de repetição
while	Criação de comando de repetição
do... while	Criação de comando de repetição
for... in	Criação de comando de repetição em um objeto
throw	Sinalização de erros
try... catch... finally	Tratamento de erros

EXCEÇÕES EM JAVASCRIPT

```
<html><head>
<title>Untitled</title></head>
<body>
<script>
 anoNascimento = prompt("Digite o seu ano de nascimento: ");
 try
 {
              if (anoNascimento <= 0)
              {  throw new Error("O ano não pode ser <= 0!"); }

             if (isNaN(parseInt(anoNascimento)))
              {  throw new Error("Não digitar letras! Digitar somente números inteiros!");  }
 } catch(e)  {
             alert(e.message);
 }
</script></body></html>
```

OBJETOS EM JAVASCRIPT

```
<html><head><title>Untitled</title></head>
<body>
<script>
var cubo = {
  medidaLados : 0,
  areaTotal   : 0,
  areaBase    : 0,
  areaLateral : 0
}
```

OBJETOS EM JAVASCRIPT

```
function Cubo(m) {
  this.medidaLados = m;
            this.areaTotal = function calculaAreaTotal() {
              x = 6 * Math.pow(this.medidaLados, 2);
                        return x
            };

            this.areaBase = function calculaAreaBase() {
              y = Math.pow(this.medidaLados, 2);
                        return y
            };

            this.areaLateral = function calculaAreaLateral() {
              z = 4 * Math.pow(this.medidaLados, 2);
                        return z
            };
};
```

OBJETOS EM JAVASCRIPT

```
var meuCubo = new Cubo(5);
alert("Area Total do Cubo = " + meuCubo.areaTotal() + "\n Area da Base do
Cubo = " + meuCubo.areaBase() + "\n Area da Base Lateral do Cubo = " +
meuCubo.areaLateral());
</script>
</body></html>
```

Duas observações referente ao exemplo acima:

1) O construtor sempre começa com a 1º letra em maiúsculo.
2) Os métodos foram criados diretamente dentro da função construtora.

ARRAY EM JAVASCRIPT

```
var vet_Numeros = new Array(123,130,156,210,434,1298);
```

Ou criamos também da seguinte forma:

```
var vet_Numeros = new Array;
vet_Numeros[0] = 123;
vet_Numeros[1] = 130;
vet_Numeros[2] = 156;
vet_Numeros[3] = 210;
vet_Numeros[4] = 434;
vet_Numeros[5] = 1298;
```

ARRAY EM JAVASCRIPT

Arrays podem conter dados do tipo objeto:

var vet_Dados = new Array(“149.432.323-33”,“Anderson”,”casado”,{idade: 51, dataNiver: “12/08/1968”},”Campinas”);

Para ler um dado do tipo objeto, fazemos:

vet_Dados[indice].idade; ou vet_Dados[4].idade;

vet_Dados[indice]. dataNiver; ou vet_Dados[4].dataNiver;

ARRAY EM JAVASCRIPT

MÉTODOS	EXPLICAÇÃO DO MÉTODO
Concat(arg1,arg2,...,argn)	Acrescenta elementos (os passados como parâmetros) a um array existente
forEach(função(elmento, índice, obj)	Percorre todos os elementos de um array
indexOf(arg)	Retorna o índice de um elemento procurado de um array
lastIndexOf(arg)	Retorna o último índice de um elemento procurado num array
join(arg)	Converte os elementos de um array numa string permitindo acrescentar caracter
pop()	Remove o último elemento de um array retornando o valor removido
push(arg)	Acrescenta um novo elemento a um array retornando a quantidade de elementos existentes no array incluindo o elemento inserido
reverse()	Inverte a odem dos elementos de um array no próprio array
sort(função)	Ordena de forma crescente ou descrescente (dependendo do parâmetro opcional) um array
toString()	Converte os elementos de um array em string

OUTROS OBJETOS EM JAVASCRIPT

OBJETOS	EXPLICAÇÃO DO OBJETO
Math	Esse objeto não tem um construtor não podendo ser usado com o operador *new*. Tem as seguintes propriedades: LN10, LN12, LOG10E, PI, SQRT. Os métodos do Math são: abs(x), max(x,y,...,n), min(x,y,...,n), round(x), floor(x), ceil(x), pow(x,y), sqrt(x), random(), exp(x), log(x), sin(x), cos(x)
Number	Esse objeto é um construtor de valores numéricos e pode ser usado com o operador new (var valor = new Number(valor)). Pode-se usar o Number() como função global usada para conversão de tipos de dados. As propriedade do Number() são: MAX_VALUE, MIN_VALUE. Seus métodos são: toString(x), toFixed(x), toPrecision(x) e toExponential(x).

Date	Trata-se de um construtor de datas e horas:
	var d = new Date();
	var d = new Date(milissegundos);
	var d = new Date(data_string);
	var d = new Date(ano, mes, dia, hora, minute, segundo, milissegundo);
	É importante lembrar que a data retornada será a data extraída do SO do computador usado.
	Os métodos do Date() são: getDay(), getFullYear(), getHours(), getMilliseconds(), getMinutes(), getMonth(), getSeconds(), getTime(), getUTCDate() – retorna o dia do mês (1-31), getUTCDay() – retorna o dia da semana (0-7), getUTCFullYear(), getUTCHours(), getUTCMilliseconds(), getUTCMinutes(), getUTCMounth(), getUTCSeconds(), e os métodos set.
	Outros métodos de conversão: toDateString(), toTimeString(), toString().

Document	O objeto document é a página atual que está sendo visualizada nesse momento. As propriedades desse objeto são: alinkColor, area, bgColor, classes, domain, fgColor, Form, ids, Image, link, location, tags, title, URL, vlinkColor.

FUNÇÕES EM JAVASCRIPT

* FUNÇÃO ESTÁTICA

```
<html><head><title>Untitled</title>
<script>
function calculaMediaAluno(n1, n2, n3, n4)  {
  return (n1 + n2 + n3 + n4)/4;  };
</script>
</head>
<body>
<script>
   var media = calculaMediaAluno(2.3, 9.4, 6.6,10);
   alert(media);
</script></body></html>
```

FUNÇÕES EM JAVASCRIPT

* FUNÇÃO DINÂMICA

```
<html><head><title>Untitled</title>
<script>
  var calculaMediaAluno = new Function("var m = (n1+n2+n3+n4);
        var media = m/4;  alert('Media do Aluno: ' + media);");
</script></head>
<body>
<button type="button" onclick="calculaMediaAluno(2.3,9.4,6.6,10);">
Calcular Media</button>
</body></html>
```

FUNÇÕES EM JAVASCRIPT

* FUNÇÃO EXPRESSÃO

```
<html><head><title>Untitled</title>
<script>
var calculaMediaAluno = function(n1,n2,n3,n4)  {
   return alert("Media do aluno: " + (n1 + n2 + n3 + n4)/4);
};
</script></head>
<body>
<button type="button" onclick="calculaMediaAluno(2.3,9.4,6.6,10);">
Calcular Media</button>
</body></html>
```

FUNÇÕES EM JAVASCRIPT

* FUNÇÃO ARROW

A nova versão do JavaScript trouxe uma nova forma de criar funções usando o operador =>.
Esta nova forma de se trabalhar com funções são chamadas Arrow Functions ou (“fat arrows”).
Ou seja, uma expressão arrow function possui uma sintaxe mais curta.

Nesse tipo de função, não precisamos mais usar:

- function
- uso de colchetes
- return (é implícito nesse tipo de função)

FUNÇÕES EM JAVASCRIPT

```
var soma = function(num1, num2) {
    return num1 + num2;
}

var soma = (num1, num2) => {
   return num1 + num2;
}

var soma = (num1, num2) => {  num1 + num2; }
```

FUNÇÕES EM JAVASCRIPT

```
var contaPalavras = function(frase) {
   return frase.split(‘ ’).length;
}

var contaPalavras = (frase)  =>  {
   return frase.split(‘ ’).length;
}

var contaPalavras = frase =>
    frase.split(‘ ’).length;
```

FUNÇÕES EM JAVASCRIPT

```
var objAluno = function(RA,nome) {
   return { ‘RA’: RA; ‘NOME’: nome;  };
};

var objAluno = (RA,nome) => {
   return { ‘RA’: RA; ‘NOME’: nome;  };
};

var objAluno = (RA,nome) =>
   ({ ‘RA’: RA; ‘NOME’: nome});
```

FUNÇÕES GLOBAIS EM JAVASCRIPT

FUNÇÕES	EXPLICAÇÃO
escape(string)	Retorna uma nova string com caracteres substituídos por sua sequência hexadecimal.
eval(string)	O argumento da função eval() é uma string. Se a string representa uma expressão, eval()avalia a expressão. Se o argumento representa uma ou mais declarações de JavaScript, eval() avalia as declarações.
isFinite(valor)	Retorna true se o valor for um número. Determina se o valor transmitido é um número finito.
isNaN(valor)	Retorna true se valor NÃO for numérico.
Number(valor)	Converte em número o valor passado como argumento da função.
parseFloat(string)	Converte em número real uma string. Caso o 1º caracter não seja um número, a função retornar NaN.
parseInt(string)	Converte em número inteiro uma string. Caso o 1º caracter não seja um número, a função retornar NaN.

STRINGS EM JAVASCRIPT

Para criarmos uma string, temos duas formas:

var frase = “Curso de Javascript”; (tipo de dado)
var frase = new String(“Curso de Javascript”);

Vejamos as propriedades do objeto String:

PROPRIEDADES	EXPLICAÇÃO
length	Retorna a quantidade de caracteres existentes na string.
prototype	Permite adicionar novas propriedades ou métodos a um objeto já criado.
constructor	Referencia a função que cria o objeto String.

Bom… chegamos ao final dos slides.

Conforme comentado no início esse ESTUDO DIRIGIDO é somente um start de nossos estudos de JAVASCRIPT.
Agora, segue um video para iniciantes interessante!

Só para você conhecer alguns outros detalhes dessa linguagem de programação.

https://www.youtube.com/watch?v=Ptbk2af68e8

OBS: não estou fazendo propaganda do Guanabara. E, nem indicando o curso dele para vocês. Trata-se apenas de uma referencia de um vídeo que pode agregar conhecimento aos nossos estudos.
Bom final de semana. Até semana que vem!