JavaScript – parte 2

Profa. Me. Andréia Rodrigues Casare
Email: casareandreia@gmail.com

Pós e pré incremento

- As operações: num++ e num—
  - aumentam e diminuem o valor da variável em 1
  - Observe o código a seguir:

```
<script>
        var num=10;
        var x;
        x=num++;
        document.write(x;
</script>
```

  - O resultado da execução é 10 e não 11 !

Pós e pré incremento

- pós incremento o valor é adicionado à variável depois
- pré incremento é adicionado antes, vamos ver nosso código novamente.
- X=num++;
- Neste caso primeiro é adicionado o valor de num a X e só depois num tem o valor incrementado, então, no momento da atribuição do valor de num a x, num ainda não foi incrementada.
- X=++num;
- O incremento é feito antes que o valor seja passado à variável, isso é chamado de pré incremento.

Try – Catch – Finally

- A instrução try “monitora” a execução dos comandos do bloco, caso haja algum erro, este erro é passado à instrução catch que executa seu bloco de comandos, caso não ocorra nenhum erro o bloco catch é ignorado, ao final são executados os comandos do bloco finally.
- Em resumo, try executa um ou mais comandos, caso haja algum erro nesta execução o bloco catch é executado, no final de tudo o bloco finally é executado.

Exemplo do uso try

```
<!doctype html>
<html lang="pt-br">
      <head>
            <title>Curso de Javascript</title>
            <meta charset="UTF-8">
            <script>
                  try{
                        alertt("CFB");
                  }catch(erro){
                        document.write("Houve um erro no bloco try<br>");
                  }finally{
                        document.write("Comando try finalizado<br>");
                  }
            </script>
      </head>
      <body>

      </body>
</html>
```

Neste bloco try será gerado um erro, pois, o comando alert está escrito de forma incorreta, então, a execução é passada ao bloco catch que executa o comando de impressão.

Alterando a mensagem de erro

- Podemos interceptar a mensagem de erro gerada pelo try, essa mensagem é passada ao catch pelo parâmetro que demos o nome de “erro” catch(erro), podemos usar outro nome para este parâmetro, mas o nome “erro” é bem sugestivo.

```
<script>
	try{
		alertt("CFB");
	}catch(erro){
		document.write(erro.message+"<br>");
	}finally{
		document.write("Comando try finalizado<br>");
	}
</script>
```

Try – Catch – Finally

- O bloco finally não é obrigatório, caso não queira executar comandos ao finalizar a instrução try basta não utilizar o bloco finally, como no código a seguir :

```
<script>
	try{
		alert("CFB");
	}catch(erro){
		document.write("Houve um erro no bloco try<br>");
		document.write("Erro gerado: "+erro.message+"<br>");
	}
</script>
```

getElementById

- Com o método **getElementById** é possível referenciar um elemento HTML e alterar o estilo CSS.
- Exemplo:

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Testando getElementById</title>
        <meta charset="UTF-8">
        <script>
            var txt;
        </script>
    </head>
    <body>
        <p id="texto">Curso de Javascript</p>
        <script>
            tx=document.getElementById("texto");
            tx.style.color="#F00";
        </script>

    </body>
</html>
```

innerHTML

- A propriedade innerHTML altera o conteúdo de uma tag HTML.
- Suponhamos que em algum momento seja preciso mudar o texto definido em parágrafo. Exemplo:

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de Javascript</title>
        <meta charset="UTF-8">
            <script>
            </script>
    </head>
    <body>
        <p id="texto">Curso de Javascript</p>
        <script>
            document.getElementById("texto").innerHTML="Fatec Itapetininga";
        </script>
    </body>
</html>
```

innerHTML

- Podemos usar a propriedade **innerHTML** para simplesmente obter o valor de um elemento ou para alterar como já vimos.
- Exemplo:

```
<!doctype html>
<html lang="pt-br">
	<head>
		<title>Curso de Javascript</title>
		<meta charset="UTF-8">
		<script>
			var tx;
		</script>
	</head>
	<body>
		<p id="texto">Curso de Javascript</p>

		<script>
			tx=document.getElementById("texto").innerHTML;
			document.write("Texto da tag p: " + tx);
		</script>
	</body>
</html>
```

getElementsByTagName

- O método **getElementsByTagName** é um método para referenciar os elementos HTML, este método retorna o(s) elemento(s) com o nome indicado, o ponto mais interessante é que caso haja várias tags iguais e isso é bastante comum, o método funciona como um vetor com todas as tags.

Exemplo uso getElementByTagName

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de Javascript</title>
        <meta charset="UTF-8">
        <script>
            var tag_1;
            var tag_2;
        </script>
    </head>
    <body>
        <p>Fatec Itapetininga</p>
        <p>Curso de Javascript</p>
        <p>www.fatecitapetininga.edu.br</p>
        <p>www.youtube.com.br</p>
        <script>
            tag_1=document.getElementsByTagName("p")[0];
            tag_2=document.getElementsByTagName("p")[1];
            document.write(tag_1.innerHTML+"<br>");
            document.write(tag_2.innerHTML+"<br>");
            tag_1=document.getElementsByTagName("p");
            document.write(tag_1[0].innerHTML+"<br>");
        </script>
    </body>
</html>
```

querySelectorAll

- O querySelectorAll é um método que retorna os elementos do documento que correspondem a uma classe de CSS, como um objeto dentro de uma lista de objetos (array).
- Uma das grandes vantagens de usar o método querySelectorAll é que podemos nos referenciar tanto a tags como classes CSS.
- Exemplos:

  1 – Obter todos os elementos com a tag <p> e armazenar os elementos no array.

```
var ps = document.querySelectorAll("p");
ps[0].style.backgroundColor="#F00";
```

querySelectorAll

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de Javascript</title>
        <meta charset="UTF-8">
        <script>
            var ps;
        </script>

    </head>
    <body>
        <p>Fatec Itapetininga</p>
        <p>Curso de Javascript</p>
        <script>
            ps = document.querySelectorAll("p");
            ps[0].style.backgroundColor="#F00";
            ps[1].style.backgroundColor="#F00";
            document.write(ps[0].innerHTML+"<br>");
            document.write(ps[1].innerHTML+"<br>");
        </script>
    </body>
</html>
```

Acessando elementos de formulários

- Para formulários existe uma maneira muito simples de acessarmos os elementos, no caso dos formulários os elementos do <form> são todos armazenados em um vetor, assim como todos os elementos do documento.
- tag=document.forms["curso"]; //Este comando permite obter todos os elementos do formulário “curso” através da variável “tag”.
- document.write(tag.elements["fNome"].value); //Com o vetor de elementos passado à variável “tag” agora podemos referenciar qualquer elemento pelo nome e obter, por exemplo, seu valor.

Exemplo

```
<!doctype html>
<html lang="pt-br">
	<head>
		<title>Curso de Javascript</title>
		<meta charset="UTF-8">
		<script>
			var tag;
		</script>
	</head>
	<body>
		<form name="curso" action="#" method="post">
			<label>Nome</label><br>
			<input type="text" name="fNome" value="Digite seu nome"><br><br>
			<label>Senha</label><br>
			<input type="password" name="fSenha"><br><br>
			<input type="submit" name="btEnviar" value="Enviar">
		</form>
		<script>
			tag=document.forms["curso"];
			document.write(tag.elements["fNome"].value);
		</script>
	</body>
</html>
```

Trabalhando com data e hora

- Em Javascript temos a nossa disposição uma classe com todos os métodos necessários para trabalhar com data e hora, é a classe “Date”, já está tudo pronto, basta chamar os métodos desejados.
- Uma observação importante é que a data e a hora retornadas pela classe Date, são a data e hora do computador do cliente e não do servidor, ou seja, o código busca a data e hora configurados em sua máquina, se seu relógio estiver errado a data mostrada será errada.
- Para trabalharmos com Data e Hora é simples, basta criar um objeto do tipo Date que já possui todos os métodos necessários para obtermos as informações da data e hora.

Trabalhando com data e hora

```
<script>
      var data=new Date();
      var dia=data.getDate();
      var mes=data.getMonth()+1;
      var ano=data.getFullYear();
      document.write(dia + "/" + mes + "/" + ano + "<br>");
</script>
```

var dia=data.getDate(); // getDate() retorna o dia do mês, neste caso armazena na variável dia.
var mes=data.getMonth()+1; // getMonth() retorna o mês, mas em forma de vetor, então, como o primeiro elemento do vetor tem índice [0], janeiro = 0, fevereiro = 1, março = 2 e assim por diante, por isso adicionamos 1 ao final do método.
var ano=data.getFullYear(); // getFullYear() retorna o ano com quatro dígitos.

Escrevendo a data por estêncil

```
<script>
      var data=new Date();
      var meses=new Array("Janeiro","Fevereiro","Março","Abril","Maio","Junho","Julho","Agosto","Setembro","Outubro","Novembro","Dezembro");
      var dia=data.getDate();
      var mes=data.getMonth();
      var ano=data.getFullYear();
      document.write("Belo Horizonte, " + dia + " de " + meses[mes] + " de " + ano + "<br>");
</script>
```

Métodos para trabalhar com data e hora

Método	Descrição	Exemplo
getDay();	Retorna o dia da semana, domingo=0, segunda=1	var diaSem=data.getDay();
getDate();	Retorna o dia do mês	var dia=data.getDate();
getMonth();	Retorna o mês, janeiro=0, fevereiro=1	var mes=data.getMonth();
getFullYear();	Retorna o ano com 4 dígitos, ex: 2015	var ano=data.getFullYear();
getHours();	Retorna a hora	var hora=data.getHours();
getMinutes();	Retorna os minutos	var minutos=data.getMinutes();
getSeconds();	Retorna os segundos	var segundos=data.getSeconds();
toDateString();	Retorna a data por estêncil, padrão EUA	var dataTexto=data.toDateString();
toDateString();	Retorna a data no formato 16/12/2015	var dataPadrao=data.toLocaleDateString();
toLocaleString();	Retorna a data e a hora 16/12/2015 23:13:00	var dataHoraTexto=data.toLocaleString();

Exercício

- Faça um script usando todos os métodos de data e hora.

Biblioteca Math

- Math é uma “biblioteca” disponível com vários métodos e constantes disponíveis para facilitar o trabalho com matemática.

Biblioteca Math – Constantes

Constante	Descrição
Math.E	Retorna o número Euler
Math.PI	Retorna o número PI
Math.SQRT2	Retorna a raiz quadrada de 2
Math.SQRT1_2	Retorna a raiz quadrada de ½
Math.LN2	Retorna o logaritmo natural de 2
Math.LN10	Retorna o logaritmo natual de 10
Math.LOG2E	Retorna base 2 do logaritmo de E
Math.LOG10E	Retorna base decimal do logaritmo de E

Biblioteca Math – Métodos

Método	Descrição
abs(x)	Retorna o valor absoluto de X
acos(x)	Retorna o arco cosseno de X
asin(x)	Retorna o arco seno de X
atan(x)	Retorna o arco tangente de X como um valor numérico entre PI/2 e PI/2 radiano
atan2(y,x)	Retorna o arco tangente do quociente dos argumentos y e x
ceil(x)	Retorna o valor de X arredondado para cima
cos(x)	Retorna o cosseno de x em radianos
exp(x)	Retorna o valor de Ex

Biblioteca Math – Métodos

floor(x)	Retorna o valor de X arredondado para baixo
log(x)	Retorna o logaritmo natural (base E) de x
max(x,y,z,...,n)	Retorna o maior valor dos argumentos
min(x,y,z,...,n)	Retorna o menor valor dos argumentos
pow(x,y)	Retorna o valor de x elevado a y
random()	Retorna um número aleatório entre 0 e 1
round(x)	Retorna o inteiro mais próximo, arredonda para cima ou para baixo
sin(x)	Retorna o seno de x em radianos
sqrt(x)	Retorna a raiz quadrada de x
tan(x)	Retorna a tangente do ângulo x

Exemplos

```
<!doctype html>
<html lang="pt-br">
      <head>
            <title>Curso de Javascript</title>
            <meta charset="UTF-8">
            <script>
                  var num=Math.random();
                  document.write(num);
            </script>
      </head>
      <body>

      </body>
</html>
```

```
<script>
      var num=Math.random()*100;
      document.write(num);
</script>
```

```
<script>
      var num=Math.floor(Math.random()*100);
      document.write(num);
</script>
```

Funções

- Uma ótima forma de controlar a execução de determinado bloco de código em um programa é utilizar funções, assim podemos criar toda uma rotina de programação e executar esta rotina em um momento específico, ter esse controle é extremamente útil e fundamental.
- É bastante simples trabalhar com funções em Javascript.
- Sintaxe:

```
function nomeDaFunção(lista de argumentos){
    comandos;
    comandos;
    retorno da função;
}
```

Funções – exemplo

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de Javascript</title>
        <meta charset="UTF-8">
        <script>
            function escreve(){
                document.write("Fatec Itapetininga<br>");
            }
            for(var i=0; i<5; i++){
                escreve();
            }
        </script>
    </head>

</html>
```

Exemplo 2 – recebendo parâmetros

```
<!doctype html>
<html lang="pt-br">
    <head>
        <title>Curso de Javascript</title>
        <meta charset="UTF-8">
            <script>
                function notas(n1, n2, n3, n4){
                    var soma=n1+n2+n3+n4;
                    if(soma >= 60){
                        document.write("<p style='color:#00F'>Aprovado</p>");
                    }else{
                        document.write("<p style='color:#F00'>Reprovado</p>");
                    }
                }
                notas(20,10,15,5);
            </script>
    </head>
    <body>
    </body>
</html>
```

Exemplo 3 – usando matriz

```
<script>
    var alunos=new Array();
    var aluno=new Array();
    var i;
    function notas(n1, n2, n3, n4){
        var soma=n1+n2+n3+n4;
        if(soma >= 60){
            document.write("<span style='color:#00F'>Aprovado</span><br>");
        }else{
            document.write("<span style='color:#F00'>Reprovado</span><br>");
        }
    }
    aluno=["Alfredo",20,10,15,5];
    alunos[0]=aluno;
    aluno=["Fabiana",20,20,10,30];
    alunos[1]=aluno;
    aluno=["Claudemir",5,5,7,2];
    alunos[2]=aluno;
    aluno=["Neuma",9,15,15,20];
    alunos[3]=aluno;
    aluno=["Julio",20,20,30,30];
    alunos[4]=aluno;
    aluno=["Clara",5,15,10,20];
    alunos[5]=aluno;
    aluno=["Adriano",15,15,25,25];
    alunos[6]=aluno;
    aluno=["Rosimeire",20,19,30,25];
    alunos[7]=aluno;
    for(i=0; i<alunos.length; i++){
        document.write("Aluno: " + alunos[i][0] + " = ");
        notas(alunos[i][1], alunos[i][2], alunos[i][3], alunos[i][4]);
    }
</script>
```

Exemplo 4 – retornando valor

```
<script>
    var alunos=new Array();
    var aluno=new Array();
    var i, nota;
    function notas(n1, n2, n3, n4){
        var soma=n1+n2+n3+n4;
        if(soma >= 60){
            document.write("<span style='color:#00F'>Aprovado</span><br>");
        }else{
            document.write("<span style='color:#F00'>Reprovado</span><br>");
        }
        return soma;
    }
    aluno=["Alfredo",20,10,15,5];
    alunos[0]=aluno;
    aluno=["Fabiana",20,20,10,30];
    alunos[1]=aluno;
    aluno=["Claudemir",5,5,7,2];
    alunos[2]=aluno;
    aluno=["Neuma",9,15,15,20];
    alunos[3]=aluno;
    aluno=["Julio",20,20,30,30];
    alunos[4]=aluno;
    aluno=["Clara",5,15,10,20];
    alunos[5]=aluno;
    aluno=["Adriano",15,15,25,25];
    alunos[6]=aluno;
    aluno=["Rosimeire",20,19,30,25];
    alunos[7]=aluno;
    for(i=0; i<alunos.length; i++){
        document.write("Aluno: " + alunos[i][0] + " = ");
        nota=notas(alunos[i][1], alunos[i][2], alunos[i][3], alunos[i][4]);
        document.write(" - Nota: " + nota + "<br>");
    }
</script>
```

Exemplo 4

```
<!doctype html>
<html lang="pt-br">
	<head>
		<title>Curso de Javascript</title>
		<meta charset="UTF-8">
		<script>
			var numeros=new Array();
			var resultado;

			function media(nums){
				var tam=nums.length;
				var soma=0;
				for(i=0; i< tam; i++){
					soma+=nums[i];
				}

				return Math.floor(soma/tam);
			}

			resultado=media([10,2,5,30,25,19]);

			document.write("Média: " + resultado);
		</script>
	</head>
	<body>

	</body>
</html>
```

Exercícios

1) Crie uma função que receba dois números inteiros e retorne a soma dos múltiplos de 3 e 5.
2) Crie uma função que receba dois números inteiros por parâmetro e retorne a soma dos N números inteiros existentes entre esses dois números (inclusive).
3) Crie uma função que receba três valores, 'a', 'b' e 'c', que são os coeficientes de uma equação do segundo grau e retorne o valor do delta, que é dado por '$b^2 - 4ac$'.