JavaScript

IFRN – INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIAS DO RIO GRANDE DO NORTE

JavaScript – Netspace/Sun Microsystems

Visão geral

A linguagem JavaScript foi desenvolvida para rodar no lado do cliente, isto é, a interpretação e o funcionamento da linguagem dependem de funcionalidades hospedadas no navegador do usuário. Isso é possível porque existe um interpretador JavaScript hospedado no navegador.

Javascript

Manipular o navegador

Com JavaScript, podemos controlar o comportamento do navegador em diversos aspectos, como criar janelas pop-up, apresentar mensagens ao usuário, alterar as dimensões do navegador, inferir na barra de status, retirar menus, fechar e abrir janelas.

Javascript

Interagir com formulários

JavaScript é capaz de acessar os campos e valores digitados em um formulário HTML e proceder à validação dos dados, realizar cálculos e fornecer dicas de preenchimento dos campos.

Interagir com outras linguagens dinâmicas

JavaScript pode ser usada em conjunto com outras linguagens para cumprir tarefas complementares relacionadas ao fluxo da programação.

Javascript

Introdução à linguagem

A linguagem JavaScript foi inventada por Brendan Eich, da Netscape, e a primeira versão da linguagem denominada JavaScript 1.0 foi introduzida no navegador Netscape 2.0 em 1996. Atualmente, o nome oficial da JavaScript é ECMAScript.

ECMA – European Computer Manufacturers Association, trata-se de uma associação fundada em 1961 dedicada à padronização de sistemas de informação.

Javascript

Caixas de diálogo

Caixas de diálogo é uma janela do tipo pop-up que se destina a fornecer informações ou cancelar dados do usuário.

A linguagem JavaScript possui três métodos (ou funções), para o objeto Windows, destinadas a criar três tipos de caixas de diálogo:

1. Caixa de alerta
2. Caixa de diálogo de confirmação
3. Caixa de diálogo para entrada de string

Javascript

Caixa de alerta – destina-se a colocar na interface do usuário uma caixa de diálogo contendo uma mensagem a ele.

```
<script type="text/javascript">
    alert("Alô Mundo!");
</script>
```

```
<script type="text/javascript">
    alert("Alô Mundo!\nCheguei.");
</script>
```

Javascript

Caixa de diálogo de confirmação – o método confirm() do objeto Windows destina-se a colocar na interface do usuário uma caixa de diálogo contendo dois botões, normalmente denominados de OK e Cancel, e, ao clicar um deles, ele confirma ou cancela uma determinada ação.

```
<script type="text/javascript">
    confirm("Você tem certeza que quer apagar o arquivo?\n
                Esta operação não poderá ser desfeita.");
</script>
```

Javascript

Caixa de diálogo para entrada de dados – o método **prompt** do objeto Windows, destina-se a colocar na interface do usuário uma caixa de diálogo contendo um campo para que ele digite uma string. Esta função admite dois argumentos, conforme mostrado a seguir:

```
<script type="text/javascript">
    prompt("Olá visitante\nInforme a data de seu nascimento.", "dd/mm/aa");
</script>
```

Javascript

Escrever HTML com JavaScript

Vimos os três métodos do objeto Window destinado a criar caixas de diálogo. Vamos examinar a seguir o método write() do objeto Document cuja finalidade é escrever marcação HTML em um documento.

```
Window.document.write("arg1");

Ou simplesmente

document.write("arg1");
```

Exemplo:

```
<script tepy="text/javascript">
      document.write("<p>Alô mundo<br />Cheguei.</p>");
</script>
```

Javascript

Atrelar um evento com JavaScript

Eventos são muito usados em JavaScript e viabilizam a interatividade em uma página Web. Na verdade, eventos viabilizam a própria existência da JavaScript. Sem eles, não teríamos como fazer funcionar os scripts.

No exemplo seguinte, será mostrado a utilização dos eventos onclick:

Javascript

Exemplo evento onclick

```
<button type="button" onclick="alert('Caixa de alerta');">Alert</button>
<button type="button" onclick="confirm('Caixa de confirmação');">Confirm</button>
<button type="button" onclick="promp('Caixa de dados');">Promp</button>
```

Javascript

No exemplo a seguir, vamos demonstrar o uso do método ***write()*** sendo chamado a partir de um evento, após a página ter sido carregada.

```
<body>
	<div>TODO write content</div>
	<h1>Exemplo com evento chamando o método write()</h1>
	<p class="xpto" id="teste">document.write() - Após carregamento da página</p>
		<p class="nav-arq" id="referencia">Arquivo exemplo: <a href="c1-onclick.html">
			&laquo; anterior</a> | <a href="c1-click.html">próximo &raquo;</a></p>
		<button type="button"
		   onclick="document.write('Conteúdo inserido após carregamento da página')">
			Inserir conteúdo aqui
		</button>
</body>
```

Javascript

Inserir JavaScript na HTML

Existem três maneiras de se inserir JavaScript em um documento HTML:

- **Inline**: inserir o script diretamente na seção body do documento. Trata-se de prática não recomendada de acordo com o princípio da separação das camadas de desenvolvimento.
- **Incorporado**: inserir o script na seção head do documento.
- **Externo**: escrever o script em um arquivo externo e inserir com um link na seção head do documento.

Javascript

Exemplo de script inline:

```
<body>
    ...
    <button type="button" onclick="alert('Olá visitante!')">
    ...
</body>
```

Incorporar script ao documento pode ser uma prática tolerável em situações muito particulares, como quando se trata de um só documento utilizando um script exclusivo.

Javascript

Exemplo de script incorporado:

```
...
<head>
    ...
    <script type="text/javascript">
        //script aqui
</head>
...
```

As modernas práticas de desenvolvimento preconizam o uso de script externos a serem linkados ao documento.

Javascript

Exemplo de script externo:

```
...
<head>
    ...
    <script type="text/javascript" src="scripts/menu_script.js"></script>
        //script aqui
</head>
...
```

Note o uso do atributo src para indicaro caminho para o arquivo que contém o script.

Javascript

Sistema léxico da JavaScript

Entende-se por sistema léxico de uma linguagem o conjunto de palavras que compõem a linguagem. Portanto, iremos estudar a **estrutura léxica da JavaScript**, bem como as regras gramaticais e sintáticas para escrever scripts.

Javascript

Tamanho da caixa

A linguagem JavaScript é sensível ao tamanho de caixa (case *sensitive*). Isso significa que nomes de varáveis, funções e demais identificadores são diferenciados quando escritos com letras maiúsculas ou minúsculas.

Javascript

Comentários

A linguagem JavaScript adminte três tipos de marcadores de comentários:

- Comentário em linha única (variante 1)
  <script type="text/javascript">
  // caixa de alerta da JavaScript
  alert("Alô mundo!"); **// caixa de alerta**
  </script>
- Comentário em linha única (variante 2)
  <script type="text/javascript">
  <!-- caixa de alerta da JavaScript
  alert("Alô mundo!"); **<!-- caixa de alerta**
  </script>
- Comentário em múltiplas linhas
  /* comentário de múltiplas linhas, começa com /* e finaliza com ***/**

Javascript

Declarações

Um script consiste em uma série de instruções escritas segundo uma sintaxe própria e rígida. As instruções, escritas em uma sequência lógica, determinam a realização de tarefas com a finalidade de obter um resultado final.

Cada uma das instruções de um scrript constitui uma declaração independente e existem duas sintaxes para separar uma declaração da outra. Separe-as com ponto-e-vírgula ou coloque cada declaração em uma linha separada. Observe a seguir:

```
//declarações na mesma linha
A = 5; b = 8; alert("Alô mundo!");

//declarações em linhas separadas
 a = 5;
 b = 8;
 alert("Alô mundo!");
```

Javascript

Espaços em branco e quebras de linha

Quebras de linhas e espaços em branco, quando inseridos entre nomes de variáveis, nomes de funções, números e entidades similares da linguagem, são ignorados na sintaxe JavaScript. Contudo, para strings e expressões regulares, tais espaçamentos são considerados. Veja os exemplos:

- As duas sintaxes a seguir são válidas:
  a=27; e a = 27;
  alert("Olá"); e alert("Olá");
  function() {... } e function () {... }

- As sintaxes a seguir causam um erro:
  a=2 7; //espaço entre os algarismos de um número não é permitido.

  document.write(" Eu sou
  uma string ") //quebra de linha em uma string.

Javascript

- A sintaxe a seguir é válidas:

```
document.write("<p> Eu sou \
    uma string</p>")                    // uso de \ para quebrar linha em string
```

Javascript

Literais

Na terminologia JavaScript, a palavra literal designa qualquer dado – valor fixo (não variável) – que se insere no script. Nos exemplos a seguir, os valores 45 e Alô Mundo! São literais:

- As duas sintaxes a seguir são válidas:

```
a = 45;
mensagem = "Alô Mundo!";
```

Javascript

Literais

Existem seis tipos de dados literais conforme descritos a seguir:

- Inteiros;
- decimais;
- Booleanos;
- Strings;
- Arrays;
- Objetos.

Javascript

Literais

Exemplo de declarações usando o literal inteiros:

```
c = 32;              // base10
d = -119;            // base10
e = 0x110B6;         // base hexadecimal
f = 0xf56a2;         // base hexadecimal
```

Exemplo de declarações usando o literal decimais:

```
a = 3.1416;
b = -76.89;
c = .33333;
d = 3E5;                    //equivale a 3x10**5 = 300000
b = -47.78965432E10;        // equivale a -47.78965432x10**10 = 477896543200
```

Javascript

Literais

Exemplo de declarações usando o literal booleano:

```
a = true;
b = false;
```

Exemplo de declarações usando o literal strings:

```
nome = "José Antônio";
outroNome = "Pedro Dantas";
```

Javascript

Literais

Pode-se obter alguma formatação básica ao escrever strings com o uso de caracteres especiais para a linguagem JavaScript. No exemplo a seguir:

```
mensagem = “Obrigado pela visita.\nVolte em breve.”;
```

Foi inserido o caractere \n na string. Ele faz com que haja o pulo de uma linha quando renderizada a string no navegador, resultando em algo como:

```
Obrigado pela visita.
Volte em breve.
```

Javascript

Tabela 1 de caracteres especiais da JavaScript

\b	Backspace - \u0008
\f	Form feed - \u000C
\n	Nova linha - \u000A
\r	Retorn do carro - \u000D
\v	Tabulação vertical - \u000B
\t	Tabulação horizontal
\'	Apóstrofo ou aspas simples - \u0027
\"	Aspas duplas - \u0022
\\	Barra invertida - \u005C
\XXX	Caractere Latin-1 expresso por dígitos octais de 1 a 377
\xxx	Caractere Latin-1 expresso por dígitos hexadecimais de 00 a FF
\uXXXX	Caractere Unicode expresso por quatro dígitos hexadecimais

Javascript

Nas três últimas linhas da tabela 1 anterior, consta a sintaxe geral para a representação de caracteres em strings nos sistemas octal e hexadecimal. O sistema octal está em desuso e, assim, deve ser evitado.

O sistema hexadecimal admite uma sintaxe para Latin-1 e uma sintaxe para Unicode. Para consultar as tabelas de caracteres, visite o site http://www.unicode.org/. Lá, você encontrará toda a codificação Unicode nos seus variados formatos.

Javascript

Para demonstrar o funcionamento da codificação, mostrarei a seguir três caracteres e seus respectivos códigos e exemplo de uso na sintaxe JavaScript.

Tabela 2 – caracteres Unicode na sintaxe JavaScript

Caractere	Octal	Latin-1 hexadecimal	Unicode hexadecimal
©(Copyright)	\251	\xA9	\u00A9
®(Marca registrada)	\256	\xAE	\u00AE
¶ (parágrafo)	\247	\xA7	\u00A7

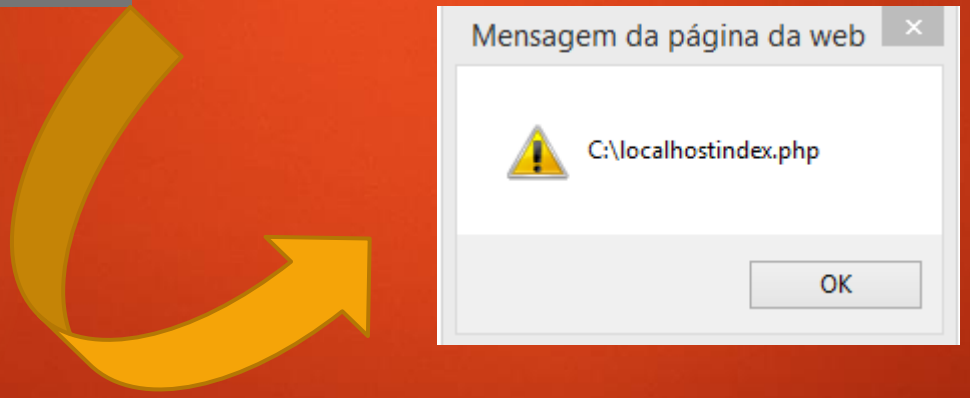
Veja o exemplo l**caractereEspeciais.html**

Javascript

Caractere de escape

A barra invertida tem um emprego específico na escrita de string da JavaScript. É conhecida como caractere de escape e usada para representar caracteres especiais como mostrado e também para escapar caracteres normalmente não permitidos dentro de uma string. O caractere de escape, barra invertida, é ignorado dentro dastring.

```
alert("C:\\localhost\index.php");
```



Javascript

Arrays

Os literais arrays, na sintaxe JavaScript, são os conjuntos de zero ou mais valores, separados por vírgula e envolvidos por colchetes ([]). Os valores contidos em um array recebem um índice sequencial começando com zero.

Exemplo:

```
frutas = [“laranja”, “pera”, “goiaba”, “morango”];
```

Para recuperar os valores de um array, usa-se a sintaxe composta pelo nome do array seguida de um índice, entre colchetes, como mostra o exemplo a seguir:

```
frutas[0] contém o valor laranja
frutas[3] contém o valor morango
```

Javascript

Arrays

Um array pode conter qualquer tipo de dado da JavaScript, incluindo expressões, objetos e outras arrays. Por exemplo:

```
arrayMisto = [“laranja”, 34, “casa”, true, [1,3,5], a+b];
```

Javascript

Objetos

Objeto é um tipo de dado constituído por uma coleção de dados, ou seja, é uma unidade que armazena dados formatados em pares nome/valor.

Segundo uma definição formal, podemos dizer que objeto é uma coleção não ordenada de propriedades e métodos constituída por pares nome/valor.

Os valores do par nome/valor podem ser tanto valores **primitivos** (números e strings) como outros **objetos**.

Javascript

Criando Objetos

Para criar um objeto vazio (sem propriedades), a sintaxe é:

```
var livro = new Object();
```

Identificamos a varável de nome **livro**, atribuída ao objeto a criar e o operador ***new*** seguido de uma função a qual denominamos **construtor** que inicia o objeto.

O construtor **Object()** é nativo da linguagem JavaScript e cria um objeto genérico vazio.

Javascript

Tabela 3 contendo alguns dados sobre um livro publicado pela editora Novatec.

1	Título	AJAX com Jquery
2	Autor	Maurício Samy Silva
3	Páginas	328
4	Preço	R$69,00
5	Frete	A calcular
6	Capítulo 1	Revisão do AJAX
7	Capitulo 2	Funções para requisições AJAX
8	Capitulo 3	Eventos e miscelânea
9	Capitulo 4	Requisições XML
10	Capitulo 5	Introdução ao formato JSON
11	Capitulo 6	Requisições JSON

Javascript

Sintaxe formal

Para definir uma propriedade do objeto livro denominada titulo, cujo valor seja a string "AJAX com jQuery", usamos a sintaxe mostada a seguir:

```
livro.titulo = "AJAX com jQuery";
```

Javascript

Observe a sintaxe para criar o objeto livro da tabela 3.:

```
var livro = new Object();
livro.titulo = "AJAX com jQuery";
livro.autor = "Maurício Samy Silva";
livro.paginas = 328;
livro.preco = "R$69,00";
livro.freteSedex = function (ceporigem, cepdestino, peso) {
    var valorFrete = "";
    //script de cálculo do frete aqui
    return valorFrete;
}
livro.capitulo1 = "Revisão do AJAX";
livro.capitulo2 = "Funções para requisições AJAX";
livro.capitulo3 = "Eventos e miscelanea";
livro.capitulo4 = "Requisições XML";
livro.capitulo5 = "Introdução ao formato JSON";
livro.capitulo6 = "Requisições JSON";
```

Javascript

Objetos

No exemplo anterior, criamos um objeto livro com dez propriedades e um método.

Com a sintaxe mostrada, podemos a qualquer momento acrescentar novas propriedades ou métodos ou alterar os já existentes. Para criar uma nova propriedade, basta declarar o seguinte:

livro.novaPropriedade = valor da nova propriedade;

Javascript

Recuperando valores de propriedades e métodos de um objeto

```
var nomeAutor = livro.autor;
var capituloCinco = livro.capitulo5;
var valorDoFrete = livro.freteSedex(59015410,69190120, 1060);

//Exibindo o resultado
Alert (“Autor: “ + nomeAutor + “\nCap5: “ + capituloCinco + “\nValor frete: “ + valorDoFrete);
```

Há uma sintaxe alternativa para recuperar os valores de propriedades de objetos que usa colchetes ([]) no lugar do operador .(ponto) conforme mostrado a seguir:

```
var nomeAutor = livro[autor];
var capituloCinco = livro[capitulo5];
```

Como exercício. Faça um script JavaScript que, use o loop for/in para retornar todos os pares nome/valor do nosso objeto livro.

Javascript


```
<script type="text/javascript">
        var livro = new Object();
        livro.titulo="AJAX com jQuery";
        livro.autor = "Maurício Samy Silva";
        livro.paginas = 328;
        livro.preco = "R$69,00";
        livro.freteSedex = function (ceporigem, cepdestino,peso) {
            var valorFrete = "";
            //script paracalcular o frete aqui
            return valorFrete;
        };
        livro.capitulo1 = "Revisão do AJAX";
        livro.capitulo2 = "Funções pararequisições AJAX";
        livro.capitulo3 = "Eventos e miscelanea";
        livro.capitulo4 = "Requisições XML";
        livro.capitulo5 = "Introdução ao formato JSON";
        livro.capitulo6 = "Requisições JSON";

        //Exibindo objeto com for/in
        var pares = "";
        for (var prop in livro) {
            pares += prop + ": " + livro[prop] + "\n";
        };
        alert(pares);
</script>
```

Javascript

Como já foi dito, um objeto é um conjunto de dados, então pode conter outros objetos como dados, ou seja, objetos podem ser aninhados. Vamos exemplificar o conceito de objetos aninhados usando uma alternativa para criar nosso objeto livro.

Vamos supor que os capítulos do livro devam estar contidos em um objeto chamado capítulos. A definição do objeto capitulo a ser aninhada ao objeto livro se faz como mostrado a seguir:

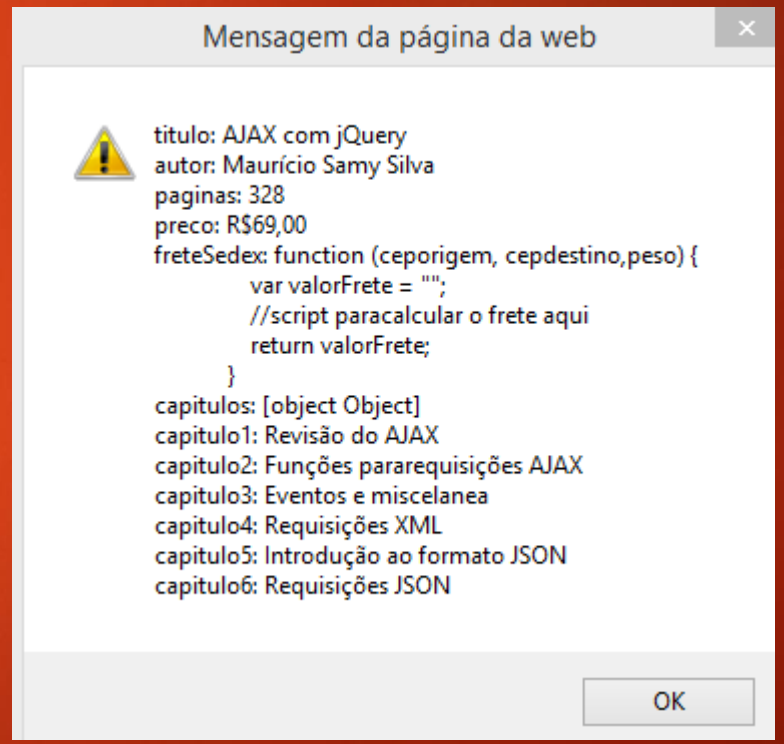
```
var livro.capítulos = new Object();
livro.capítulos.capitulo1 = “Revisão do AJAX”;
livro.capítulos.capitulo2 = “Funções para requisições AJAX”;
...
livro.capítulos.capitulo10 = “Requisições JSON”;
```

Javascript

Exercício 1:

1. modifique o objeto livro, de forma que, os capítulos façam parte de um objeto capítulos.
2. Exiba em uma tela (alert()), todas os membros do livro, inclusive os capítulos.

resultado



Javascript


```
<script type="text/javascript">
        var livro = new Object();
        livro.titulo="AJAX com jQuery";
        livro.autor = "Maurício Samy Silva";
        livro.paginas = 328;
        livro.preco = "R$69,00";
        livro.freteSedex = function (ceporigem, cepdestino,peso) {
            var valorFrete = "";
            //script paracalcular o frete aqui
            return valorFrete;
        };
        livro.capitulos = new Object();
        livro.capitulos.capitulo1 = "Revisão do AJAX";
        livro.capitulos.capitulo2 = "Funções pararequisições AJAX";
        livro.capitulos.capitulo3 = "Eventos e miscelanea";
        livro.capitulos.capitulo4 = "Requisições XML";
        livro.capitulos.capitulo5 = "Introdução ao formato JSON";
        livro.capitulos.capitulo6 = "Requisições JSON";

        alert(msg);
...
```

Javascript

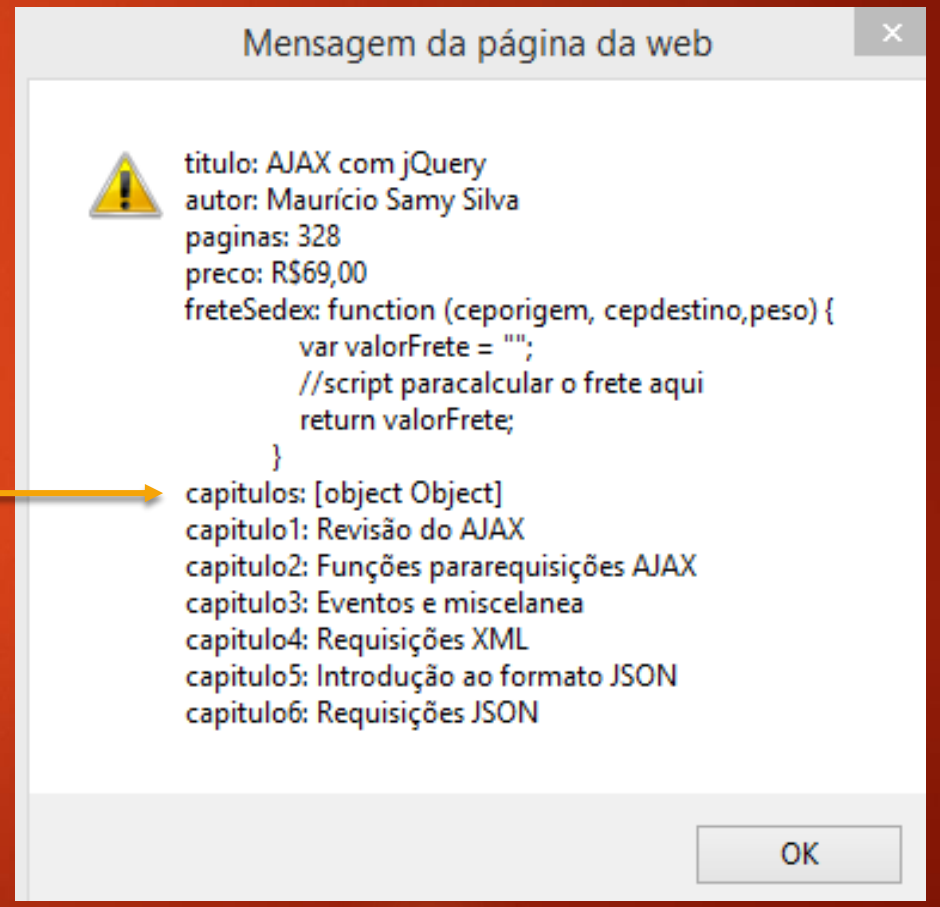

```
//Exibindo objeto com for/in
        var msg = "";
       for (var prop in livro) {
          msg += prop + ": " + livro[prop]+"\n";
          if (typeof livro[prop] == "object"){
             for (var cap in livro[prop]){
                msg += cap + ": " + livro[prop][cap] + "\n";
             }
          }
       }

        alert(msg);

</script>
```

Javascript

Exercício 2:

1. O que devo fazer para eliminar a linha 9 do resultado. Veja a figura abaixo.

Linha 9



Javascript

Basta colocar o if antes da varaível msg.

```
//Exibindo objeto com for/in
var msg = "";
for (var prop in livro) {
	if (typeof livro[prop] == "object"){
		for (var cap in livro[prop]){
			msg += cap + ": " + livro[prop][cap] + "\n";
		};
	} else {
			msg += prop + ": " + livro[prop]+"\n";
	};
 }

alert(msg);
```

Javascript

Construtor

Nos exemplos anteriores, criamos o objeto livro como o uso do operador ***new*** e de uma função construtora denominada ***Object()***.

Denomina-se função construtora ou simplesmente construtor uma função capaz de criar objetos. Não estamos limitados a criar objetos como o uso de construtores nativos, pois podemos criar nossos construtores pernsonalizados.

Javascript

Construtor

Vamos supor que em nosso script precisamos manipular cilindros e resolvemos criar um construtor para gerar cilindros.

```
1. function Cilindro(r,h) {
2.       this.raioBase = r;
3.       this.altura = h;
4. };
5. cilindroUm = new Cilindro(2,5);
6. alert("Raio da base: " +  cilindroUm.raioBase + "\nAltura: " + cilindroUm.altura);
```

Linha 1: função construtora Cilindro que adminite dois argumentos (raio=r, e altura = h)
Linha 2: a palavra-chave nativa da linguagem, this que faz referencia ao novo objeto criado
Linha 5: com o construtor Cilindro cria-se novo objeto, denominado cilindroUM, com raio=2, e altura=5

Javascript

Criar um construtor para cilindros com a finalidade de retornar os argumentos passados à função não tem valor prático algum. Vamos melhorar nosso construtor acrescentando alguns métodos.

Conhecendo-se o raio da base (r) e a altura (h) de um cilindro circular reto, é possível determinar as seguintes grandezas do cilindro:

1. Área da base: área do circulo de raio r: areaBase = πr^2
2. Área lateral: área da superfície lateral: areaLateral = $2\pi rh$
3. Área total: soma da área lateal com a área das bases: areaTotal = $2\pi rh + 2\pi r^2$
4. Volume: área da base vezes a altura: volume = $\pi r^2 h$

Como exercício, crie métodos para o construtor Cilindro capazes de retornar as grandezas listadas.

Javascript


```
<script type="text/javascript">
        function Cilindro (r,h) {
            this.raioBase = r;
            this.altura = h;
            this.areaBase = function calculaAreaBase() {
                aBase = Math.PI * Math.pow(this.raioBase,2);
                return aBase;
            };
            this.areaLateral = function calculaAreaLateral(){
                aLateral = 2 * Math.PI * this.raioBase * this.altura;
                return aLateral;
            };
            this.areaTotal = function calculaAreaTotal(){
                return 2 * aBase + aLateral;
            };
            this.volume = function calculaVoluma(){
                return aBase * this.altura;
            };
        }
        cilindroUm = new Cilindro(3,10);
            alert("Raio da base: " + cilindroUm.raioBase + "\nAltura: " + cilindroUm.altura +
                    "\nÁrea Base: " + cilindroUm.areaBase() + "\nÁrea Lateral: " + cilindroUm.areaLateral() +
                    "\nÁrea total: " + cilindroUm.areaTotal() + "\nVolume: " + cilindroUm.volume());
</script>
```

Javascript

Categorias de objetos

Os objetos da linguagem podem ser agrupados em três categorias, a saber:

- **Objetos nativos**: aqueles próprios da linguagem;
- **Objetos do ambiente de hospedagem**: aqueles próprios do dispositivo que interpreta a libguagem (um navegador gráfico, por exemplo);
- **Objetos customizados**: aqueles criados pelo desenvolvedor.

Javascript

Variável, tipo de dado e conversão

```
// Variável
var tw = "TreinaWeb";

// Redeclarando a variável
var tw = 20;
```

Javascript

Aritmética

```
var t = 10;
var w = 6;
var tw = 0;

// Soma
tw = t + w;

// Subtração
tw = t - w;

// Multiplicação
tw = t * w;
// Divisão
tw = t / w;

// Módulo
tw = t % w;
```

Javascript

Aritmética - Problemas com arredondamento

```
// Definição das variáveis
var t = 0.7 - 0.6; // Retorna: 0.09999999999999998
var w = 0.2 - 0.1; // Retorna: 0.1

// Comparações de igualdade:
console.log("Primeiro teste: "+(t == w)); // False
console.log("Segundo teste: "+(t == 0.1)); // False
console.log("Terceiro teste: "+(w == 0.1)); // True
console.log("Quarto teste: "+(t == 0.09999999999999998)); // True
```

Javascript

Objetos

```
// Criando um objeto
var objeto = {
    tw : "TreinaWeb Cursos"
};
// Exibindo o valor da propriedade 'tw' do objeto 'objeto'
// utilizando ponto (.)
document.write("O retorno de 'objeto.tw' é: "+objeto.tw);
document.write("<br />");

// Exibindo o valor da propriedade 'tw' do objeto 'objeto'
// utilizando colchetes ([])
document.write("O retorno de 'objeto[\"tw\"]' é: "+objeto["tw"]);
```

Javascript

Criando objeto (exemplo 2)

```
// Criando um objeto
var objeto = {
    tw : "TreinaWeb"
};

document.write("O valor de 'tw' é: "+objeto.tw+"<br />"); // O valor da propriedade 'tw'

objeto.tw = "TreinaWeb Cursos"; // Configurando a propriedade 'tw'

document.write("O valor de 'tw' agora é: "+objeto.tw+"<br />"); // O valor de 'tw' após a configuração

// Primeiro utilizando um identificador. Para isso utilize o ponto (.) // Criando duas novas propriedades
objeto.x = 6;

// Agora utilizando uma string literal. Para isso utilize os colchetes ([])
objeto["y"] = 4;
// Exibindo o valor das novas propriedades
document.write("O valor da nova propriedade 'x' é: "+objeto.x+"<br />");
document.write("O valor da nova propriedade 'y' é: "+objeto.y);
```

Javascript

O operador DELETE remove uma propriedade do objeto:

```
// Criando um objeto
var obj = {
    prop_1 : 1,
    prop_2 : 2,
    prop_3 : 3
};
// Exibindo os valores do objeto 'obj'
document.write("O valor de 'prop_1' do objeto 'obj' é: "+obj.prop_1+"<br />");
document.write("O valor de 'prop_2' do objeto 'obj' é: "+obj.prop_2+"<br />");
document.write("O valor de 'prop_3' do objeto 'obj' é: "+obj.prop_3+"<br />");
document.write("<br />");

// Agora será excluída a propriedade 'prop_2' do objeto 'obj'
delete obj.prop_2;
// Os valores após deletar 'porp_2' são:
document.write("O valor de 'prop_1' do objeto 'obj' agora é: "+obj.prop_1+"<br />");
document.write("O valor de 'prop_2' do objeto 'obj' agora é: "+obj.prop_2+"<br />");
document.write("O valor de 'prop_3' do objeto 'obj' agora é: "+obj.prop_3+"<br />");
```

Javascript

O laço for—In para percorrer um objeto

```
// Criando um objeto
var obj = {
    x : 5,
    y : 2,
    z : 7
};

// Testando se toString é enumerável
document.write("toString em 'obj' é enumerável? "+obj.propertyIsEnumerable("toString")+"<br />");
document.write("Mas toString existe em 'obj'? "+("toString" in obj)+"<br />");

// Iterando no objeto 'obj' e exibindo os nomes das propriedades
document.write("Nome das propriedades enumeráveis do objeto 'obj': <br />");
for(prop in obj){
    document.write(prop+"<br />");
}
```

Javascript

DOM -

Document Object Model – **DOM**, é uma API utilizada para representar e manipular o conteúdo dos documentos.

No DOM, os elementos aninhados de um documento HTML são representados como uma árvore de objetos.

A representação em árvore de um documento HTML contém nós representando marcações, ou elementos HTML, como <body> e <p>, e nós representando string de texto.

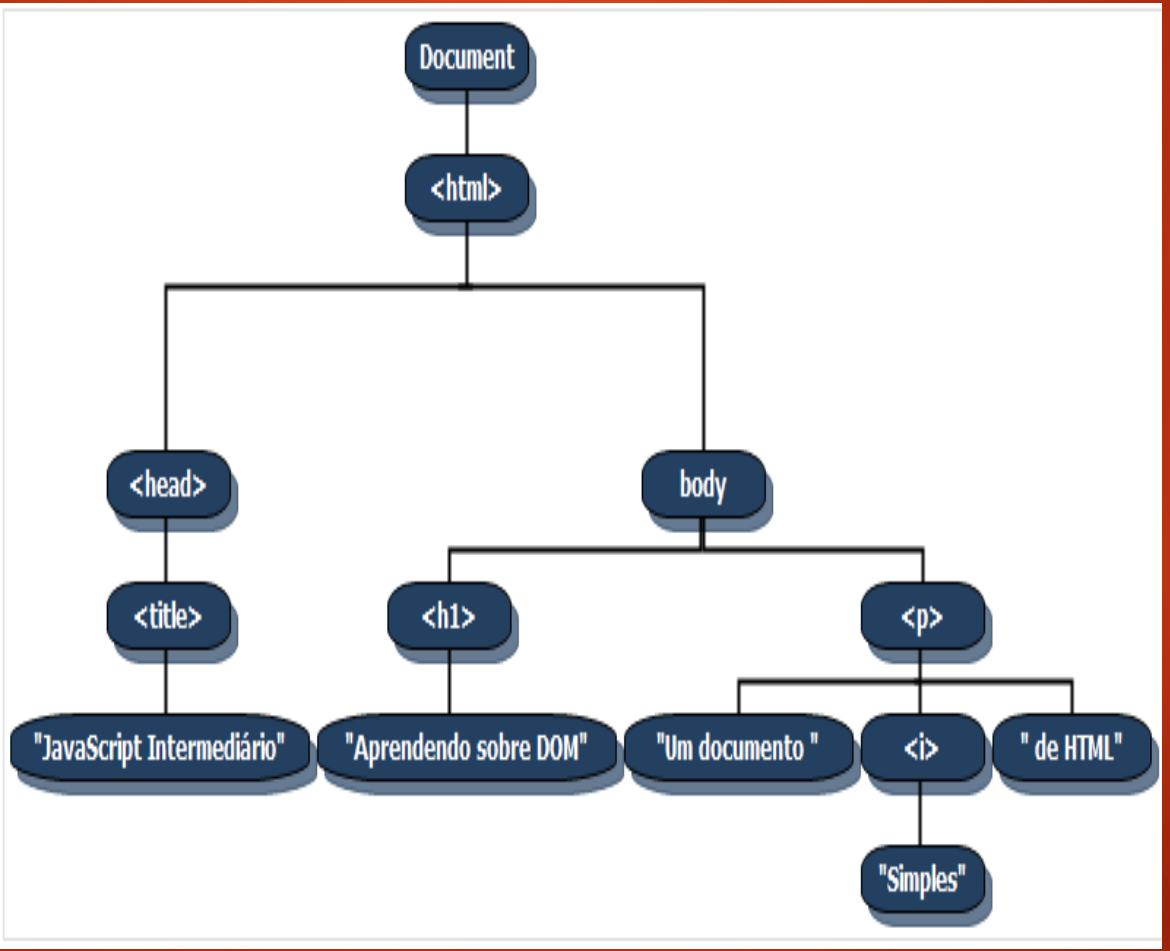
Javascript

DOM -

```
<html>
   <head>
      <title>JavaScript Intermediário</title>
   </head>
   <body>
      <h1>Aprendendo sobre DOM</h1>
      <p>Um documento <i>simples<i> de HTML</p>
   </body>
</html>
```

Javascript

A representação DOM do documento anterior é a árvore a seguir:



Javascript

Expressões Regulares

Uma expressão regular é um objeto que descreve um padrão de caracteres. A classe RegExp de JavaScript representa as expressões regulares, e tanto String quanto RegExp definem métodos que utilizam expressões regulares para executar funções poderosas de comparação de padrões e de localização e substituição de texto.

Javascript

Os objetos RegExp podem ser criados com a construtora RegExp().

```
var expRegular = new RegExp("s$");
```

Mas são criados mais frequentemente com o uso de uma sintaxe literal:

```
var expRegular = “/s$/”;
```

Javascript

Método exec() de RegExp, que executa uma expressão regular numa string especificada.

```
// Criando uma variável com um texto para análise
var texto = "TreinaWeb Cursos - JavaScript Avançado!"

// Criando uma expressão regular que retornará um array
var expRegularTrue = new RegExp("TreinaWeb");

// Criando uma expressão regular que retornará null
var expRegularFalse = new RegExp("treinaweb");

// Executando test()
// Retorno é um array
console.log(expRegularTrue.exec(texto));

// Retorno igual a null
console.log(expRegularFalse.exec(texto));
```