Programação Web

Javascript

Ivo Calado

Instituto Federal de Educação, Ciência e Tecnologia de Alagoas

22 de Fevereiro de 2016

Roteiro

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

O que vimos até aqui?

- Linguagens que possuem como foco a visualização da informação
- Não possibilitam a execução de fluxos de código

O que vimos até aqui?

- Linguagens que possuem como foco a visualização da informação
- Não possibilitam a execução de fluxos de código
- Não são destinadas a programadores!!

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

O que vimos até aqui?

- Linguagens que possuem como foco a visualização da informação
- Não possibilitam a execução de fluxos de código
- Não são destinadas a programadores!!
- JavaScript foi projetado para adicionar interatividade à páginas HTML
- Trata-se de uma linguagem de script
- Trata-se de uma linguagem interpretada

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

O que vimos até aqui?

- Linguagens que possuem como foco a visualização da informação
- Não possibilitam a execução de fluxos de código
- Não são destinadas a programadores!!
- JavaScript foi projetado para adicionar interatividade à páginas HTML
- Trata-se de uma linguagem de script
- Trata-se de uma linguagem interpretada

O que é uma linguagem interpretada?

O que vimos até aqui?

- Linguagens que possuem como foco a visualização da informação
- Não possibilitam a execução de fluxos de código
- Não são destinadas a programadores!!
- JavaScript foi projetado para adicionar interatividade à páginas HTML
- Trata-se de uma linguagem de script
- Trata-se de uma linguagem interpretada

O que é uma linguagem interpretada?

Significa que o script é executada sem uma prévia compilação

Curiosidades

Qual a relação entre Java e JavaScript?

Curiosidades

Qual a relação entre Java e JavaScript?

Nenhuma. Trata-se de linguagens completamente diferentes. Uma criada pela Sun e a outra pela Netscape!

Curiosidades

Qual a relação entre Java e JavaScript?

Nenhuma. Trata-se de linguagens completamente diferentes. Uma criada pela Sun e a outra pela Netscape!

Qual o nome oficial da linguagem JavaScript?

ECMAScript! Desenvolvido e mantido pela *ECMA International Organization*

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Hello World

Para adicionar algum script JavaScript basta fazer uso da tag **script** tendo como valor do campo **type**: **text/javascript**

```
<html>
  <body>
    <script type="text/javascript">
      document.write("<h1>Hello World!</h1>");
    </script>
  </body>
</html>
```

document.write é a maneira padrão para escrita na saída!

Como tornar JavaScript compatível com navegadores antigos?

- Deve-se fazer uso dos comentários HTML para "esconder" o código JavaScript do navegador para manter a retrocompatibilidade!

```
<html>
  <body>
    <script type="text/javascript">
    <!--
      document.write("Hello World!");
    //-->
    </script>
  </body>
</html>
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

No HEAD?

- Scripts colocados na seção HEAD não são executados imediatamente e devem ser localizados em funções!

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event"
    );
}
</script>
</head>
<body onload="message()">
</body>
</html>
```

No Body?

- Se você não deseja que seu script esteja em uma função ou que ele seja destinado a escrever conteúdo na tela então coloque-o na seção **body**!

```
<html>
  <head></head>
  <body>
    <script type="text/javascript">
    document.write("This message is written by
        JavaScript");
   </script>
  </body>
</html>
```

Usando em um arquivo externo?

- Se você deseja compartilhar seu código JavaScript entre diversas páginas, pode-se fazer uso de uma página externa

```
<html>
  <head>
    <script type="text/javascript" src="xxx.js"></
        script>
  </head>
  <body onload="helloworld()">
  </body>
</html>
```

Uso do ponto e vírgula

- Em JavaScript, é facultativo o uso do ; (ponto e vírgula) para delimitar o fim de um comando!

```
<html>
  <body>
    <script type="text/javascript">
    document.write("Fim de linha COM ponto e virgula");
    document.write("Fim de linha SEM ponto e virgula")
   </script>
  </body>
</html>
```

Pulando linha em JavaScript

- Como você deve ter percebido no exemplo anterior, as duas frases ficaram na mesma linha
- Este comportamento acontece porque toda saída de enviada pelo JavaScript é tratata como código HTML. Por isso, deve-se fazer uso do "
" ao final da frase

```
document.write("Fim de linha COM ponto e virgula<br
    />");
document.write("Fim de linha SEM ponto e virgula<br
    />")
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Comentários de linhas e multilinhas

- JavaScript oferece dois tipos de comentários "//" e "/* */"
- A semântica é a mesma do que acontece em linguagens como **Java** e **C++**

```
<html>
<body>
<script type="text/javascript">
/*
The code below will write
   one heading
*/
document.write("<h1>This is a heading</h1>");
</script>
</body>
</html>
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Usando variáveis

- Variáveis em JavaScript tem a mesma semântica que em outras linguagens, isto é, armazenar informações na memória
- Assim como o restante do JavaScript, são **case-sensitive**
- Podem ser iniciadas por [aA-zZ], _ ou $
- A partir do segundo caracter poder ser dos seguintes tipos [aA-zZ], [0-9], _ ou $

Exemplo do uso de variáveis

```
<html>
  <body>
    <script type="text/javascript">
      var firstname;
      firstname="Hege";
      document.write(firstname);
      document.write("<br />");
      firstname="Tove";
      document.write(firstname);
    </script>
  </body>
</html>
```

Uso de variáveis não declaradas

- É possível atribuir valores à variáveis ainda não declaradas.

```
<html>
  <body>
    <script type="text/javascript">
      firstname="Hege";
    </script>
  </body>
</html>
```

- Nestes caso, a variável é criada no momento da atribuição!

Arrays

- Em JavaScript é possível criar array para armazenamento de diversos tipos de objetos
- Os objetos não precisam ser do mesmo tipo
- Existem 4 tipos de instanciação!

```
var myCars=new Array();
var myCars=new Array(10);
var myCars=new Array("Saab","Volvo","BMW");
var myCars=["Saab","Volvo","BMW"];
```

Indexando arrays

- Assim como Java, arrays em JavaScript são indexados a partir da posição 0

```
var myCars=["Saab","Volvo","BMW"];
document.write(myCars[0]) // Imprime Saab
```

Unindo arrays

- Em JavaScript é possível unir dois arrays de maneira bastante simples

```
<script type="text/javascript">
var parents = ["Jani", "Tove"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(children);
document.write(family);
</script>
```

Unindo todos elementos de um array em uma string

É possível unir todos os elementos de um array em uma única string através do método **join**

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.join() + "<br />"); //Utiliza o
    separador padrão ","

document.write(fruits.join("+") + "<br />"); //Utiliza
    o separador passado por parâmetro

document.write(fruits.join(" and ")); //Utiliza o
    separador passado por parâmetro

</script>
```

Exercício

Pesquisar e criar exemplos para os seguintes métodos de um array:

- push
- pop
- reverse
- shift
- slice
- sort
- unshift (se possível, testar no Firefox e IE)

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

O objeto String

Em JavaScript o objeto String possui as seguintes propriedades/métodos

- length
- toUpperCase()
- big()
- small()
- bold()
- italics()
- fixed()
- strike()
- etc

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Exercício para casa!

Pesquisar a utilização dos objetos:

- Date
- Boolean
- Math
- RegExp

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Operadores aritméticos

- JavaScript oferece 7 operadores aritméticos (+, -, *, /, %, ++, –)

Operadores aritméticos

- JavaScript oferece 7 operadores aritméticos (+, -, *, /, %, ++, –)
- Além disso, são oferecidos os operadores de atribuição correspondentes (=, +=, -=. *=, /= e %=)

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Concatenação de strings

- O operador "+" pode ser utilizado também no processo de concatenação de strings
- Se pelo menos um dos operandos for uma string então será realizado a concatenação

```
v1 = 20
v2 = "11"
v3 = v1 + v2
```

Irá resultar na string **2011** e não no número **31**!!!

Operadores de comparação

- JavaScript especifica 7 operadores de comparação: ==, ===, !=, >, <, >=, <= e

```
<script type="text/javascript">
  x = 5
  println(x == 5)
  println(x == "5")
  println(x === 5)
  println(x === "5")
</script>
```

Operadores de comparação

- JavaScript especifica 7 operadores de comparação: **==**, **===**, **!=**, **>**, **<**, **>=**, **<=** e

```
<script type="text/javascript">
  x = 5
  println(x == 5)
  println(x == "5")
  println(x === 5)
  println(x === "5")
</script>
```

- 3 operadores lógicos: **&&**, **||**, **!**

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Operadores ternário

Assim como Java e C++, em JavaScript é possível fazer uso do operador ternário.

Qual o resultado da variável **valor** após a execução do seguinte script?

```
<script type="text/javascript">
  x = 5
  valor = (x != 5)? "string de teste" : 10
</script>
```

Expressões condicionais: if-else

- A utilização da estrutura de seleção **if-else** é semelhante à Java

```
if (condition)
{
  statement 1;
  statement 2;
} // Os dois comandos são executados
if (condition)
  statement 1;
  statement 2;
  statement 3;
// Apenas o primeiro comando é executado
```

Exemplo if-else

```
<html>
<body>
<script type="text/javascript">
        var d = new Date();
        var time = d.getHours();
        if (time < 12)
        {
                document.write("<b>Bom dia</b>");
        } else
                document.write("<b>Boa tarde</b>");
</script>
</body>
</html>
```

Exercício

Criar uma página HTML que possua um script JavaScript que faz uso da função **Math.random()** para gerar números aleatórios e caso o valor for maior que 0.5 criar o link para a página do Google. Caso contrário, redirecionar para a página globo.com .

Exercício II

- Altere o exemplo **if.html** para que, dependendo da hora obtida no sistema seja impressa a aula atual (1ª, 2ª, 3ª, intervalo, 4ª, 5ª, 6ª).
- Sabe-se que o método **getMinutes()** do objeto Date retorna os minutos atuais e **getHours()** retorna a hora atual.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

O comando For

- Em JavaScript, a estrutura for segue a mesma sintaxe da linguagem Java

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
  //codigo a ser executado
}
```

O comando For

- Em JavaScript, a estrutura for segue a mesma sintaxe da linguagem Java

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
  //codigo a ser executado
}
```

- As estruturas **while** e **do-while** seguem a mesma estrutura da linguagem JavaScript

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Exercício

- Implemente uma página Web que escreva os 6 tipos de cabeçalhos fazendo uso do comando **for**

Controles de fluxo adicionais

- break, continue e **for..in**

Qual a diferença entre **break** e **continue**

Controles de fluxo adicionais

- break, continue e **for..in**

Qual a diferença entre **break** e **continue**

continue interrompe a iteração atual e inicia na próxima iteração.
break interrompe todo o loop.

for..in

- A estrutura **for..in** tem a mesma semântica do **for** extendido do Java
- Visa realizar iteração sobre arrays e listas

```
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (x in mycars)
{
        document.write(mycars[x] + "<br />");
}
```

Caixa de diálogo e Ciclo de vida das variáveis

Caixas Popup

JavaScript oferece três tipos de caixas de diálogo

- **AlertBox:** tem como objetivo informar ao usuário alguma mensagem
- **ConfirmBox:** recebe uma entrada do usuário a partir da confirmação
- **PromptBox:** recebe uma entrada do usuário a partir de uma entrada de texto

Alert Box

```
<head>
<script type="text/javascript">
function show_alert()
{
alert("Hello! I am an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="show_alert()" value="Show
alert box" />
</body>
```

Confirm Box

```
<html> <head>
<script type="text/javascript">
   function show_confirm() {
     var r=confirm("Press a button!");
     if (r==true) {
       alert("You pressed OK!");
     } else {
       alert("You pressed Cancel!");
     }
}
</script> </head>
<body>
    <input type="button" onclick="show_confirm()" value=
        "Show a confirm box" />
</body> </html>
```

Prompt Box

```
<html>
<head>
<script type="text/javascript">
function disp_prompt() {
  var fname=prompt("Please enter your name:","Your name
      ")
  alert(fname);
}
</script>
</head>
<body>
<input type="button" onclick="disp_prompt()" value="
    Display
a prompt box" />
</body> </html>
```

Funções

- Funções em JavaScript tem o mesmo propósito que em linguagens como Java e C++
- Possibilitam o reúso de código
- Funções podem ser definidas no **HEAD**, **BODY** ou num arquivo externo **.js**, porém recomenda-se não adicionar na seção BODY

```
function nome_da_funcao(var1,var2,...,varX)
{
  some code
}
```

Ciclo de vida de variáveis em JavaScript

- Se você declara uma variável dentro de uma função, ela terá escopo apenas **local** e quando a função for finalizada a variável será destruída
- Contudo, variáveis criadas fora de funções podem ser visualizadas de qualquer parte da página, chamadas variáveis **globais** e existem desde o momento que a página é carregada até quando ela for fechada

Eventos

Eventos

- Eventos são ações que podem ser detectadas em JavaScript
- A partir da implementação de eventos, podemos criar páginas dinâmicas
- Cada elemento em HTML têm um próprio conjunto de eventos que podem ser capturados
- A especificação dos eventos que serão "escutados" é definido nas tags HTML

Exemplos de Eventos

- Clique do mouse
- Carregamento de uma página Web ou imagem
- Mover o mouse sobre uma certa área da página Web
- Selecionar um campo de entrada de dados em um formulário
- Submeter um formulário

Exemplos de Eventos

- Clique do mouse
- Carregamento de uma página Web ou imagem
- Mover o mouse sobre uma certa área da página Web
- Selecionar um campo de entrada de dados em um formulário
- Submeter um formulário

Eventos são normalmente usados em combinações com funções e, portanto, a função não será executado até que o evento seja lançado

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Eventos *OnLoad* e *OnUnload*

- Esta classe de eventos são lançados no carregamento e no fechamento de uma página HTML
- OnLoad geralmente é utilizado para checar o tipo de browser utilizado ou a versão do navegador
- Um outro uso seria setar *Cookies* quando o usuário entra ou sai da página Web

Eventos OnFocus, OnBlur e OnChange

- OnFocus e OnBlur são eventos complementares. O primeiro é lançado quando um elemento ganha foco e o segundo quando perde foco
- Onchange, por sua vez, é lançado apenas se o conteúdo do elemento tiver sido alterado

Evento OnSubmit

- Evento lançado antes de um fomulário ser enviado
- Geralmente usado para fazer a validação dos campos do formulário

```
<form method="post" action="destino.htm" onsubmit="
    return checkForm()">
```

Eventos OnMouseOver e OnMouseOut

- Controlam quando o mouse entra e sai de um componente respectivamente

```
<a href="http://www.google.com.br/" onMouseOver="mouse
    (); return true">google</a>
<a href="http://g1.globo.com/" onMouseOut="mouse2();
    return true">G1</a>
```

Evento OnClick

- Evento lançado quando o usuário clica em um componente visível da tela
- Um exemplo de utilização é em botões de submissão. Caso a função JavaScript retorne falso o formulário não é submetido

```
<input type="submit" onclick="return checkForm();"
    value="Submeter" id="input3"/>
```

Tratamento de Exceções

Tratamento de exceções

- Assim como outras linguagens de programação, JavaScript oferece suporte ao tratamento de erros através do uso de exceções
- A sintaxe em JavaScript é bastante semelhante ao que acontece em C++ e Java

```
try {
    // Bloco que pode lancar exceção
} catch(ex) {
    //Tratamento de exceção
    str = ex.message //recupera o erro gerado
}
```

A cláusula **throw**

- Além de capturar exceções, podemos lançá-las
- Para tal, fazemos uso da cláusula throw
- Ela possui a mesma semântica do comando análogo, em Java, throw!

```
if(x>10)
{
  throw "Err1";
} else
   if(x<0) {
  throw "Err2";
}
```

Exercício! :)

- Pesquisar sobre o objeto **navigator**

Qual a função deste objeto?

Exercício! :)

- Pesquisar sobre o objeto **navigator**

Qual a função deste objeto?

O objeto **navigator** contém todas as informações sobre o browser visitante, como:

- Nome da aplicação
- Versão do navegador
- Plataforma
- etc

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Exercício 2! :)

- Pesquisar sobre eventos relacionados a tempo
- **setTimeout** e **clearTimeout**

HTML DOM

O que é?

O que é?

O que é?

O que é?

De acordo com o W3C o DOM (Documento Object Model) é definido como:
"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

O que é?

O que é?

De acordo com o W3C o DOM (Documento Object Model) é definido como:
"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

Em resumo HTML DOM == HTML + DOM

Em outras palavras...

- Trata-se de um padrão de modelagem dos elementos HTML como objetos
- Uma abordagem padrão para acesso e alteração do HTMI programaçõ
- Um padrão W3C

Em outras palavras: **O HTML DOM é um padrão para recuperação, alteração, adição ou remoção de elementos HTML e suas propriedades**

Nós DOM

O DOM segue a mesma filosofia de modelagem do JDOM, ou seja:

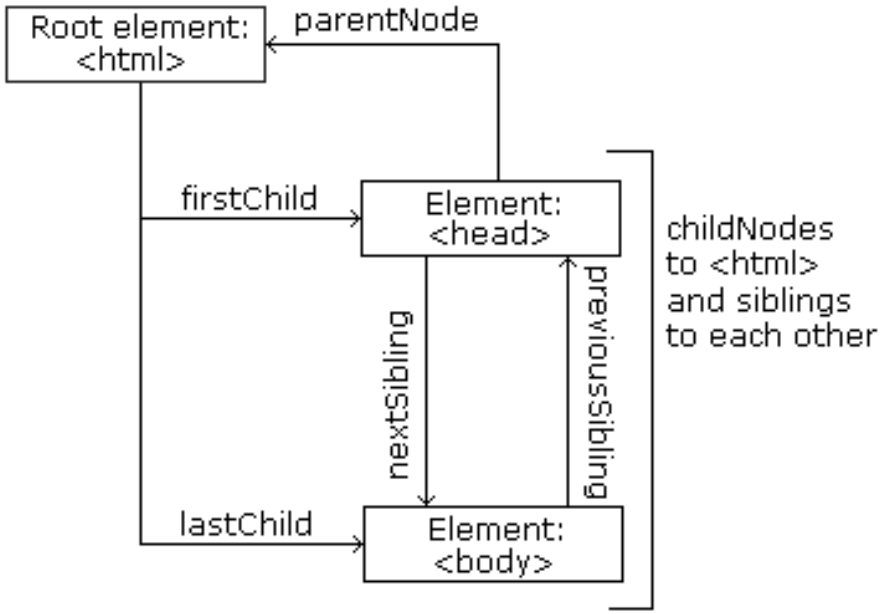
- O documento inteiro é um nó
- Cada elemento HTML é um nó
- O texto dos elementos HTML são nós
- Cada atributo do HTML é um atributo de um nó
- Comentários são nós de comentário

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Exemplo

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

Navegando entre a hierarquia de nós



Propriedades básicas de um elemento HTML DOM

- x.innerHTML - conteúdo textual de x
- x.nodeName - o nome de x
- x.nodeValue - o valor do x
- x.parentNode - referência ao elemento pai de x
- x.childNodes - Nós filhos de x
- x.attributes - nós atributos de x

A propriedade **innerHTML**

- É utilizado para recuperar ou substituir o conteúdo de um elemento HTML, inclusive <**html**> e <**body**>
- Pode ser utilizada também para visualizar o código fonte de uma página alterada dinamicamente

```
<p id="intro">Hello World!</p>

<script type="text/javascript">
  txt=document.getElementById("intro").innerHTML;
  document.write("<p>The text from the intro paragraph:
      " + txt + "</p>");
</script>
```

Propriedades childNodes e nodeValue

Uma segunda forma de obtenção é a partir da combinação das propriedades childNodes e nodeValue

```
<p id="intro">Hello World!</p>

<script type="text/javascript">
  txt=document.getElementById("intro").childNodes[0].
      nodeValue;
  document.write("<p>The text from the intro paragraph:
       " + txt + "</p>");
</script>
```

Acessando Nós

Existem basicamente três forma de acessar os nós da árvore HTML:

- **getElementById**
- **getElementByTagName**
- Navegando na estrutura dos nós

getElementById

- Trata-se do método mais utilizado para recuperação de valores
- Recebe como o id do objeto a ser acessado

Ver exemplo...

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

getElementsByTagName

- Retorna todos os elementos da tag passada por parâmetro

```
document.getElementsByTagName("p"); // Retorna todos os elementos p do HTML
```

getElementsByTagName

- Retorna todos os elementos da tag passada por parâmetro

```
document.getElementsByTagName("p"); // Retorna todos os elementos p do HTML
```

Qual o significado do código abaixo?

```
document.getElementById('main').getElementsByTagName("p");
```

getElementsByTagName

- Retorna todos os elementos da tag passada por parâmetro

```
document.getElementsByTagName("p"); // Retorna todos os elementos p do HTML
```

Qual o significado do código abaixo?

```
document.getElementById('main').getElementsByTagName("p");
```

- Os itens podem são indexados como em um vetor!!

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Exercício

Criar um formulário que antes de ser submetido será checado por valores vazio. E caso isso ocorra, cancele a transmissão e imprima um texto na cor vermelha indicando que houve erro.

Alterando as propriedades de um elemento

- Até agora vimos como alterar o conteúdo de um elemento HTML via propriedade **innerHTML**
- Porém, é possível alterar o conteúdo diretamente sem "reescrever" um novo HTML

```
<html>
<body>
<script type="text/javascript">
  document.body.bgColor="lavender";
</script>
</body>
</html>
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Exemplo 2

```
 <html>
<body>
<p id="p1">Hello World!</p>
<script type="text/javascript">
document.getElementById("p1").innerHTML="New text!";
</script>
</body>
</html>
```

Mudando um elemento a partir de eventos

- É possível alterar uma propriedade diretamente a partir do lançamento de um evento

```
<html>
<body>
<input type="button" onclick="document.body.bgColor='
    lavender';"
value="Change background color" />
</body>
</html>
```

JavaScript + CSS

- Em JavaScript as regras CSS podem ser customizadas durante a execução
- Para isso, faz-se uso da propriedade **style** de cada elemento

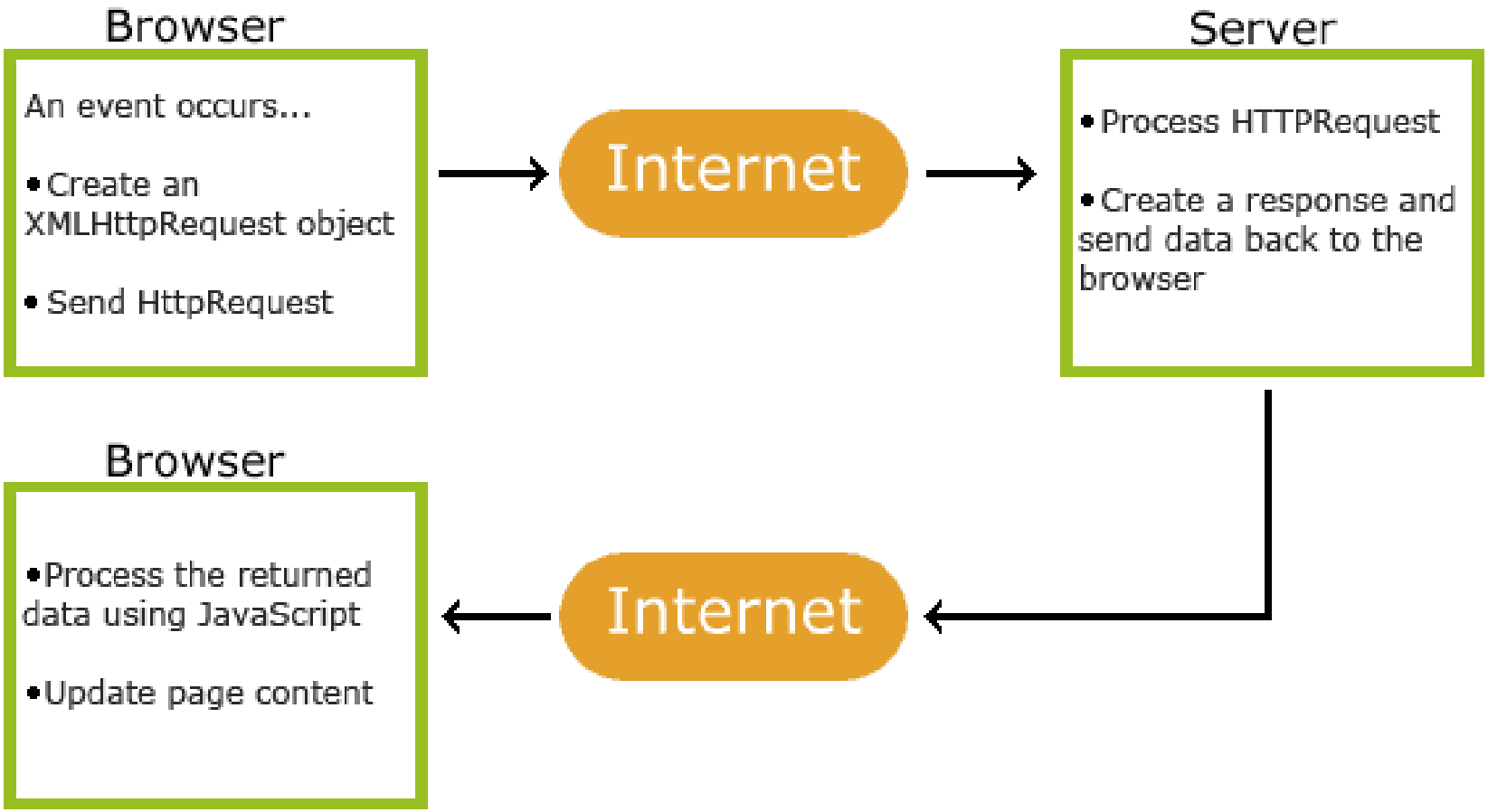
```
<script type="text/javascript">
function ChangeStyle()
{
  document.getElementById("p1").style.color="blue";
  document.getElementById("p1").style.fontFamily="Arial
      ";
}
</script>
```

Exercício

Altere o exercício anterior para que as propriedades sejam customizadas via CSS

O que é AJAX

- *Asynchronous JavaScript and XML*
- É uma técnica para criar páginas web de maneira rápida e dinâmicas
- Possibilita que as páginas Web sejam atualizadas assincronamente pela troca de pequenos pedaços de informação entre o servidor
- Torna possível a atualização de pedaços da página Web, sem a necessidade de atualização da página inteira
- Páginas Web antigas por outro lado devem recarregar a página inteira se alguma parte precisar ser atualizada
- Exemplos de aplicações com AJAX: Google Maps, Gmail, Youtube, Facebook etc


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

AJAX é baseado em padrões da Web, e utiliza uma combinação de:

- Objeto XMLHttpRequest (utilizado para a troca assíncrona de dados com o servidor)
- JavaScript/DOM (utilizado para exibir/interagir com a informação)
- CSS (utilizado para formatar a informação)
- XML (geralmente utilizado como formato de troca de informações entre cliente e servidor

O objeto XMLHttpRequest

- Todos os navegarores modernos suportam o objeto XMLHttpRequest (IE5 e IE6 usam um objeto semelhante ActiveXObject)
- O objeto XMLHttpRequest é utilizado para a troca de dados entre o servidor e o cliente

Como criar um objeto XMLHttpRequest?

$variable = newXMLHttpRequest();$

Como criar um objeto ActiveXObject?

$variable = newActiveXObject(\text{“}Microsoft.XMLHTTP\text{”});$

Como manter compatibilidade entre navegadores novos e antigos?

```
var xmlhttp;
if (window.XMLHttpRequest) {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
} else {// code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Como enviar uma requisição para o servidor

Para enviar dados para um servidor, utilizamos os métodos *open()* e *send()* do objeto XMLHttpRequest

GET

```
xmlhttp.open("GET","demo_get.asp",false);
xmlhttp.send()
```

GET com atributos

```
xmlhttp.open("GET","demo_get2.asp?fname=Henry&lname
    =Ford",false);
xmlhttp.send();
```

Como enviar uma requisição para o servidor

POST com atributos

```
xmlhttp.open("POST","ajax_test.asp",true);
xmlhttp.setRequestHeader("Content-type","
    application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Para que serve o terceiro parâmetro (bool)?

- É utilizado para indicar se a requisição será assíncrona (**true**) ou síncrona (**false**)
- Requisições síncronas fazem com que o script fique congelado até obter uma resposta do servidor. **Normalmente não é utilizado por causar travamento momentâneo do cliente**
- Requisições assíncronas são recomendadas pois:
  - Possibilitam que outros scripts sejam executados enquanto aguarda a resposta do servidor
  - Possibilita a indicação de uma função que será invocada quando a informação estiver disponível

Async = false

```
xmlhttp.open("GET","http://localhost:8080/
    ExemploHora/gethora",false);
xmlhttp.send();
document.getElementById("test").innerHTML=xmlhttp
    .responseText;
```

Async = true

```
xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status ==
        200) {
        document.getElementById("test").innerHTML =
            xmlhttp.responseText;
    }
}

xmlhttp.open("GET", "http://localhost:8080/ExemploHora/
    gethora", true);
xmlhttp.send();
```

Pesquisar como funciona a chamada **xmlhttp.responseXML**

O evento onreadystatechange

- O evento onreadystatechange é disparado toda vez que o status da requisição muda
- O status da requisição é armazenado no atributo readyState do objeto XMLHttpRequest e pode possuir os seguintes valores:
  - 0: requisição não inicializada
  - 1: conexão com o servidor estabelecida
  - 2: requisição recebida
  - 3: requisição em processamento
  - 4: requisição finalizada e resposta pronta
- A variável *status* armazena o resultado final da resposta recebida:
  - 200: Resposta recebida com sucesso
  - 404: Página não encontrada