# HTML

- Hypertext Markup Language
  - Draft – 1991; 1.1 – 1992;  2.0 – 1993;  3.0 – 1995; 4.0 – 1997; 4.01 – 1999;  5.0 (2014)
- Linguagem de marcação básica utilizada na WEB, que consiste de elementos envolvidos pelos sinais de **<** e **>**.
- O conjunto **<elementName>** é conhecido como tag
- O navegador é capaz de ler páginas HMTL e exibir seu conteúdo. O conteúdo é aquele existente entre a tag inicial e a final
- As tags não são exibidas pelo navegador, mas são utilizadas para interpretar o conteúdo das mesmas
- Cada elemento possui atributos que o qualificam e cada tag aberta deve ser devidamente fechada.
- Exemplo
  - `<elementName atributename1="attributeValue1" ... />`
  - `<elementName atributename1="attributeValue1" ... >`

```
    contentAppearAtBrowser
</elementName>
```

# HTML Tags

- A linguagem apresenta um conjunto de tags
- Uma tag não reconhecida pelo navegador é exibida como texto
- Categorias de Tags
  - Tags básicas de uma página
    - `<html>, <head>, <body>`
  - Headings (Cabeçalhos)
    - `<h1>, <h2>, <h3>, <h4>, <h5>, <h6>`
  - Parágrafos
    - `<p>, <br>`
  - Links (Âncoras)
    - `<a>`
  - Imagens
    - `<img>`
  - Comentários
    - `<!- This a comment -->`

# HTML Tags

- Categorias de Tags
  - Formatação **de Texto**
    - **<b>, <big>, , <i>, <small>, , <sub>, <sup>,...**
  - Listas (Numeradas)
    - **<ul>, <ol>**
  - Listas (Marcadores)
    - **<ul>, <ol>, <li>**
  - Tabelas
    - **<table>, <tbody>, <thead>, <th>, <tr>, td**

# HTML Tags

- Categorias de Tags
  - `<ul>, <ol>, <li>`
  - `<table>, <tbody>, <thead>, <th>, <tr>, td`
  - `<!- This a comment -->`

# HTML Tags

- A linguagem apresenta um conjunto de tags
- Uma tag não reconhecida pelo navegador é exibida como texto
- Exemplos de tags
  - `<html>, <head>, <body>`
  - `<h1>, <h2>, <h3>, <h4>, <h5>, <h6>`
  - `<p>, <br>`
  - `<a>`
  - `<b>, <big>, <em>, <i>, <small>, <strong>, <sub>, <sup>,...`
  - `<img>`
  - `<ul>, <ol>, <li>`
  - `<table>, <tbody>, <thead>, <th>, <tr>, td`
  - `<!- This a comment -->`

# Documento HTML

```
<html>
<head>
<title>Title of the document</title>
</head>

<body>
Este é o conteudo visível da página
</body>

</html>
```

# CSS (Cascade Style Sheets)

- Permite a definição de estilos que definem como os elementos HTML serão mostrados
- Desta forma é possível aplicar um determinado estilo a vários elementos, simplificando o processo de formatação dos mesmos
- Originalmente a linguagem HTML não continha atributos para diferentes fontes e cores. A introdução na versão 3.2 trouxe ônus a dificuldade para formatar os elementos. Os estilos foram adicionados na versão 4.0
- Normalmente os estilos são definidos em um arquivo à parte (.css) e o mesmo é incluído em um conjunto de páginas
- Para uma referência na linguagem consulte os endereços abaixo
  - https://www.w3schools.com/cssref/default.asp
- Um tutorial pode ser encontrado neste endereço:
  - https://www.w3schools.com/css/default.asp

# CSS (Cascade Style Sheets) Sintaxe

- Cada estilo é visto como uma regra de formatação
- A regra contém duas partes:
  - Seletor
  - Uma ou mais declarações
    - A declaração consiste de um par (propriedade:valor) finalizado por ";"
    - O conjunto de declarações deve estar entre chaves
    - É possível quebrar a declaração em várias linhas a fim de facilitar a leitura
  - Exemplo
    - `Selector {propertyName1:value1; ...; propertyNameN:valueN;}`

# CSS (Cascade Style Sheets) Sintaxe do Seletor e Uso

- O seletor pode ser utilizado de diferentes formas
  - HtmlTag
    - Neste caso o seletor é igual a alguma tag HTML e neste caso todos os elementos com aquela tag conterão a mesma formatação
  - #selectorName
    - Neste caso o nome do seletor inicia-se pelo caractere “#”. O objetivo é aplicar este estilo a um único elemento.
    - Para selecionar o elemento é é utilizado o atributo id (id="selectorName") em uma tag

    ```
    <p id="selectorName">
    ```
  - .className
    - Neste caso o nome do seletor inicia-se pelo caractere “.”
    - A classe é aplicada a um grupo de elementos
    - É possível indicar uma classe somente para determinados elementos. Desta forma somente aquelas tags serão afetas pelo estilo.

      Neste caso o seletor possuirá o seguinte formato: **`tagname.selectorName`**
    - Para selecionar o elemento é utilizado o atributo class (class="selectorName")

    ```
    <p class="selectorName">
    ```

# CSS (Cascade Style Sheets) Associação Estilo e Página

- Existem três formas de aplicar os estilos
  - Arquivo externo com estilos

```
<head>
  <link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

  - Declaração interna dos estilos

```
<head>
  <style type="text/css">
    hr {color:green;}
    p {margin-left:20px;}
    body {background-image:url("img/logo.gif");}
  </style>
</head>
```

  - Declaração no atributo “style”

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

- A primeira forma é a mais eficiente e utilizada pois os estilos poderão ser compartilhados por um grande número de páginas, facilitando manutenções no web site

# CSS (Cascade Style Sheets)
# Uso dos Estilos

- Os estilos são aplicados em cascata do mais externo para o mais interno
- A ordem de aplicação dos estilos é a seguinte:
  1. Padrão do Navegador
  2. Estilo definido em um arquivo externo
  3. Estilo definido na seção <head>
  4. Estilo definido dentro do elemento com o atributo "style"
- Os valores mais internos possuem prioridade, logo o atributo "style" fará a sobreposição nos outros estilos

# CSS
# Exemplo Básico

```
<html>
<head>
<style type="text/css">
p {color:red;text-align:center;}
h1
{
  color:blue;
  text-align:center;
}
</style>
</head>
<body>
  <p>Hello World! - Style p</p>
  <p>This paragraph is styled with CSS - Style p.</p>
  <h1>Heading - htyle h1 </h1>
</body>
</html>
```

# CSS
# Outro Exemplo - Sobreposição

```
<html>
<head>
<style type="text/css">
p {color:red;}
h1
{ color:blue;
  text-align:center;
}
</style>
</head>
<body>
  <p>Hello World! - Style p</p>
  <p style="color:blue;margin-left:20px">This paragraph is styled
with CSS - Style p.</p>
  <p>An orginal paragraph with the style defined at the head</p>
  <h1>Heading - htyle h1 </h1>
</body>
</html>
```

# CSS
# Composição de Estilos

- O termo classe para o seletor pode provocar a visão de que conceitos de programação orientada a objetos estão disponíveis, o que não é fato
- A linguagem oferece algumas formas de composição de estilos
- Composição na hieraquia dos elementos
  - Neste caso as propriedades aplicadas a elementos mais externos sejam propagadas para elementos mais internos
  - Desta forma é possível que elementos mais internos (filhos) herdem as propriedades de elementos mais externos (pai)
- Composição no atributo “class”
  - É possível compor o conteúdo de várias classes e aplicá-las simultaneamente a um elemento. Neste caso as declarações de ambas serão adicionadas
- Composição na definição da classe
  - Mais de um seletor tipo do class é escolhido para um conjunto de declarações
  - Neste caso todos compartilharão as mesmas propriedades

# CSS
# Composição na Hierarquia

- No exemplo abaixo a composição utilizou a hierarquia entre os elementos  e <p> no documento
- O valor "inherit"  para a propriedade color não é obrigatório. Sendo que para a maioria das propriedades este é o padrão utilizado, logo não é necessário repetir a propriedade na declaração da classe

```
<html>
<head>
<style type="text/css">
.foo {background-color: white; color: blue;}
.bar {background-color: yellow; color: inherit; font-weight: bold;}
</style>
</head>
<body>
<div class="foo">
  <p> Paragraph only with .foo class</p>
  <p class="bar">Paragraph with .foo + .bar class</p>
  <p>Paragraph again only with .foo class</p>
</div>
</body>
</html>
```

# CSS
# Composição no atributo “class”

- No exemplo a composição no elemento <p> a composição de estilos utilizando mais de uma classe no valor do atributo “class”.

```
<html>
<head>
<style type="text/css">
.foo {background-color:white; color:blue;}
.bar {background-color:yellow;font-weight:bold;}
</style>
</head>
<body>
<div>
  <p > Paragraph wiht no css class</p>
  <p class="foo bar">Paragraph with .foo + .bar class</p>
  <p class="foo">Paragraph again only with .foo class</p>
</div>
</body>
</html>
```

# CSS
# Composição na definição da classe

- No exemplo a composição é feita na definição onde mais de uma classe compartilha as mesmas declarações.

```
<html>
<head>
<style type="text/css">
.foo, .bar {color:blue;}
.bar {background-color:yellow;font-weight:bold;}
</style>
</head>
<body>
<div>
  <p > Paragraph wiht no css class</p>
  <p class="bar">Paragraph with .foo + .bar class</p>
  <p class="foo">Paragraph again only with .foo class<!p>
</div>
</body>
</html>
```

# CSS Topics

- Backgrounds
- Text
- Fonts
- Links
- List
- Box Model
- Border
- Margin
- Padding

CSS Box Model

# JavaScript (JS)

- JavaScript é uma linguagem de programação suportada pelos navegadores.
- A versão inicial surgiu em 1996 (Netscape) e a última versão é a ECMAScript 2021 (ES2021 – 12th edition)
- Atualmente mais de 97% dos websites utilizam Javascript no lado cliente
- Seu objetivo é melhorar a experiência do usuário e incorporar interatividade do lado cliente
- A linguagem é interpretada e o código é embutido diretamente em páginas HTML sendo executado pelo navegador (cliente).
- O navegador possui uma aplicação (javascript engine) responsável pela interpretação e execução do código

Scripting engine conformance

| Scripting engine | Reference application(s) | Conformance[46] | | | |
|---|---|---|---|---|---|
| | | ES5[47] | ES6 (2015)[48] | ES7 (2016)[49] | Newer (2017+)[49][50] |
| SpiderMonkey | Firefox 94 | 100% | 98% | 100% | 100% |
| V8 | Google Chrome 95, Microsoft Edge 95, Opera 80 | 100% | 98% | 100% | 100% |
| JavaScriptCore | Safari 15 | 100% | 99% | 100% | 90% |

- Utilizada por Webmasters e Web designers
- JavaScript não possui nenhuma relação com a linguagem Java
- Linguagem suporta:
  - Leitura e Escrita de texto em uma página HTML
  - Criar respostas a eventos ocorridos em uma página HTML
  - Validar dados informados em uma página HTML
  - Suporte a cookies

# JavaScript (JS)

- Existem diversas bibliotecas baseadas em JavaScript, entre elas:
  - JQuery (DOM, Event, CSS Animation, AJAX, UI); Angular (UI, MVC, MVVM); React (MVC, MVVM) e VUE (MVVM)
    - Model-View-Controller (MVC)
    - Model-View-View-Model (MVVM)
  - https://en.wikipedia.org/wiki/Comparison_of_JavaScript-based_web_frameworks
- Node.js é um ambiente de execução de código baseado em JS e permite a execução de código JS no lado servidor (back-end) de aplicações
  - Neste caso não é necessário o Browser
  - Aplicações baseadas em Node.js são executadas de forma concorrente no servidor
- Maiores informações:
  - Tutorial - https://www.w3schools.com/js/default.asp
  - Referência - https://www.w3schools.com/jsref/default.asp

# JavaScript
# Uso

- Para inserir um código HTML é necessário utilizar a tag <script>
  - Código interno à página

```
<script type="text/javascript">
```

  - Código em arquivo externo à página, facilitando seu reuso

```
<script type="text/javascript" src="xxx.js"></script>
```

- O código pode estar presente tanto no elemento <body> quanto no elemento <head>
  - No elemento <body> o código é executado assim que a página é carregada. O código pode estar em qualquer ponto da página, mas o ideal é que o mesmo fique no final da página
  - No elemento <head> o código é executado no momento em que um evento ocorre. Neste elemento normalmente são colocadas todas as funções

# JavaScript
# Conceitos Gerais

- Um código Javascript consiste de uma sequencia de comandos executados pelo navegador
- A linguagem é “case sensitive”
- Comando
  - Um comando pode ser finalizado por um caractere “;” e por uma quebra de linha
- Blocos
  - Uma sequencia de comandos pode ser agrupadas em blocos que neste caso iniciam-se pelo caractere “{“ e finalizado pelo caractere “}”
  - O bloco é utilizado em funções ou em estrutura de controle da linguagem
- Comentários

```
// Comentário em uma linha simples
/*  Comentário
     Em várias linhas */
```

# JavaScript Variáveis

- Variáveis
  - A linguagem é fracamente tipada e portando uma varíavel pode conter qualquer tipo de dado
  - O nome de variável de começar com uma letra com um um caractere "_"
  - Caso a variável seja declarada mais de uma vez seu valor não é alterado. Inicialmente a variável não possui conteúdo
  - Variáveis declaradas dentro de uma função são locais (escopo função) e fora são globais (escopo página)
  - É possível atribuir valor a uma variável ainda não declarada
  - Exemplo

```
var x; //declaração
var itemName="Produto";   //declaração e atribuição
globalvar;          //variável global sem o uso da keyword "var"
```

# JavaScript Operadores

- Aritméticos

| Operador | Descrição | Exemplo | Resultado | Contexto |
|---|---|---|---|---|
| + | Adição | x=y+2 | x=7 | y=5 |
| - | Subtração | x=y-2 | x=3 | y=5 |
| * | Multiplicação | x=y*2 | x=10 | y=5 |
| / | Divisão | x=y/2 | x=2.5 | y=5 |
| % | Módulo (resto da divisão inteira) | x=y%2 | x=1 | y=5 |
| ++ | Incremento | x=++y<br>x=y++ | x=6<br>x=5 | y=6 |
| -- | Decremento | x=--y<br>x=y-- | x=4<br>x=5 | y=4 |

- Atribuição

| Operador | Exemplo | Equivalente | Resultado |
|---|---|---|---|
| = | x=y | | x=5 |
| += | x+=y | x=x+y | x=15 |
| -= | x-=y | x=x-y | x=5 |
| *= | x*=y | x=x*y | x=50 |
| /= | x/=y | x=x/y | x=2 |
| %= | x%=y | x=x%y | x=0 |

# JavaScript Operadores

- Relacionais
  - É possível concatenar strings
  - Se um número for adicionado a uma string o resultado será uma string

| Operador | Descrição | Exemplo |
|---|---|---|
| == | igualdade | x==8 is false |
| === | exatamente igual (valor e tipo) | x===5 is true<br>x==="5" is false |
| != | Diferente | x!=8 is true |
| > | Maior | x>8 is false |
| < | Menor | x<8 is true |
| >= | Menor ou igual | x>=8 is false |
| <= | Maior ou igual | x<=8 is true |

- Lógicos

| Operador | Descrição | Exemplo |
|---|---|---|
| && | and lógico | (x < 10 && y > 1) is true |
| \|\| | or lógico | (x==5 \|\| y==5) is false |
| ! | not lógico | !(x==y) is true |
| ? | Operador condicional | varivalue=(condicao)?value1:value2 |

# JavaScript Objetos

- A linguagem Javascript é orientada a objetos e a linguagem possui um conjunto de classes pré-definidas com seus métodos
- Os objetos possuem propriedades e métodos
- Acesso a uma propriedade
  - objectInstance.property
- Acesso a um método
  - objectInstance.methodName();
- Todos os elementos de uma página podem ser modelados como objetos
  - Objeto document
  - Representa uma página carregada no navegador
- Outras classes
  - String
  - Date
  - Array
  - Boolean
  - Math
  - RegExp

# Javascript
# Funções

- Possui a seguinte sintaxe

```
function nomeFuncao(var1,var2,...,varX)
{
  //codigo da funcao
  return returnValue;
}
```

- Uma função pode ou não retornar valores
- As funções são utilizadas normalmente para responder a eventos

# Javascript
# Eventos

- Os eventos permitem criar páginas ou comportamentos dinâmicos
- Os eventos são detectados pela linguagem e são disparados pelos elementos de uma página HTML
- Exemplos de eventos
  - Produzidos pelo mouse (click, movimento, etc) e teclado (pressionar)
  - Carregamento de uma página ou imagem
  - Seleção de um campo em um formulario (<form>)
  - Submissão de uma página (request)
- Eventos dos elementos
  - onLoad e onUnload
  - onFocus, onBlur e onChange
  - onSubmit
  - onMouseOver

# Javascript
# Funções - Exemplo

```
<html>
<head>
<script type="text/javascript">
function displaymessage(){
  alert("Hello World!"); //mostra mensagem alerta
}
</script>
</head>
<body>
<form>
<input type="button" value="Click aqui!"
onclick="displaymessage()" />
</form>
<p>Ao pressionar o botao a funcao sera invocada</p>
</body>
</html>
```

# Javascript
# Classes e Objetos

- Javascript  é uma linguagem orientada a objetos, porém a abordagem para definir é essencialmente diferente de linguagens como C++ e Java
- Um objeto em Javascript é visto como um dicionário ou seja, uma coleção de propriedades e métodos onde a chave para acesso é o nome do método ou propriedade
- A título de comparação um objeto seria semelhante a uma **struct** da linguagem C, porém é possível associar métodos nesta estrutura
- Desta forma não existe uma sintaxe para definição de uma classe, mas sim a definição de funções e um relacionamento entre as mesmas
- Na linguagem não existe distinção entre funções e objetos. Uma função é um objeto com o código associado ao mesmo
- Em Javascript a definição de objetos envolve:
  - Definição do Construtor
  - Definição de Métodos
  - Criação de objetos

# Javascript
# Classes e Objetos - Construtor

- A definição do construtor consiste na definição de uma função, onde o nome da função equivale ao nome da “classe” que será definida
- A palavra reservada **this** é responsável por associar a propriedade ou método à classe
- É possível realizar a sobrecarga de construtores, porém isto não é realizado de forma direta como em C++ ou Java
- Exemplo de Construtor

```
function person(name,lastname,age,ecolor)
{
  this.firstname=name;
  this.lastname=lastname;
  this.eyecolor=ecolor;
}
```

# Javascript
# Classes e Objetos - Métodos

- Um método é uma função que é associada a uma classe
- Para adicionar o método a uma classe a propriedade recebe o nome da função
- A função que define o método pode realizar o acesso às propriedades da classe utilizando a palavra reservada **this**
- É possível realizar a sobrecarga de métodos, porém isto não é realizado de forma direta como em C++ ou Java
- Exemplo de Método

```
function showNameMethod(msg){
  alert(msg + " "+ this.firstname);
}
function person(name,lastname,age,eyecolor){
  this.firstname=name;
  this.lastname=lastname;
  this.age=age;
  this.eyecolor=eyecolor;
  this.showName=showNameMethod //associação do método
}
```

# Javascript
# Classes e Objetos - Criação

- A linguagem oferece algumas forma para a criação de um objeto
- Criação de uma instancia de objeto
  - Neste não foi definida uma classe de objetos, apenas um objeto

```
personObj=new Object();
personObj.firstname="Flavio";
personObj.lastname="Silva";
document.write(personObj.firstname); //acesso a propriedade
```

- Criação literal de um objeto
  - Neste não foi definida uma classe de objetos, apenas um objeto que no fundo é um dicionário

```
pObj1={firstname:"Flavio",lastname:"Silva"};
```

- Criação de um objeto a partir do construtor

```
var pObj2= new person("Flavio","Silva","green");
```

# Javascript
# Classes e Objetos - Exemplo

```
<html>
<body>
<script type="text/javascript">
function showNameMethod(msg){
  alert(msg+" "+this.firstname);
}
function person(fname,lname,ecolor){  //defines a person constructor
  this.firstname=fname;
  this.lastname=lname;
  this.eyecolor=ecolor;
  this.showName=showNameMethod;
}
var pObj= new person("Flavio","Silva","green");
for (x in pObj){
  document.write(pObj[x] + " ");
}
pObj.showName("Name:");
document.write("<h1>Writing Using JavaScript</h1>");
document.write(pObj.eyecolor);
</script>
</body>
</html>
```

# JavaScript Arrays

- Criação
  - Um array pode ser criado de três formas
  - Construtor new

```
var myCars=new Array(); // criacao
myCars[0]="Saab";       // atribuicao
myCars[1]="Volvo";
myCars[2]="BMW";
```

  - Construtor new com parâmetros

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

  - Declaração Literal

```
var myCars=["Saab","Volvo","BMW"]
```

- Acesso
  - `var element = arrayVarName[arrayIndex];`
- Modificação
  - `arrayVarName[arrayIndex] = newValue;`

# JavaScript
# Estrutura de Seleção (if-else)

- O comando if-else possui a seguinte sintaxe

```
if (condicao)
{
  //codigo executado se condicao é true
}
else
{
  //codigo executado se condicao é false
}
```

# JavaScript
# Estrutura de Seleção (if-else)

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time < 10) {
  document.write("<b>Good morning</b>");
}
else {
 document.write("<b>Good afternoon</b>");
}
</script>
<p>Demonstracao do comando if-else.</p>

</body>
</html>
```

# JavaScript
# Estrutura de Seleção (switch)

- O comando **switch** possui a mostrada abaixo e permite a escolhe de um entre vários blocos de código
- O valor de n é comparado com o valor contido em cada “case” e caso sejam iguais o bloco é executado.

```
switch(n){
case 1:
  //executa bloco 1
  break;
case 2:
  //executa bloco 2
  break;
default:
  //codigo executado independente do valor de n
}
```

# JavaScript
# Estrutura de Controle(while)

- While

```
while (condicao)
{
  //código excutado
}
```

- Exemplo

```
var i=0;
while (i<=5)
{
  document.write("valor: " + i);
  document.write("<br />");
  i++;
 }
```

# JavaScript
# Estrutura de Controle(do-while)

- do-while

```
do
{
  //bloco executado
}
while (condicao);
```

- Exemplo

```
var i=0;
do
{
  document.write(" valor:" + i);
  document.write("<br />");
  i++;
}
while (i<=5);
```

# JavaScript
# Estrutura de Controle(for)

- for

```
for (countInit;countComparacao;contIncremento)
{
  //bloco executado
};
```

- Exemplo

```
var i=0;
for (i=0;i<=5;i++){
  document.write(" valor: " + i);
  document.write("<br />");
}
```

# JavaScript
# Estrutura de Controle(for..in)

- Este método interage sobre as propriedades de um objeto ou de um array

```
for (variavel in object)
{
  code to be executed
}
```

- Exemplo

```
<html>
<body>
<script type="text/javascript">
var person={fname:"John",lname:"Doe",age:25};
for (x in person){
  document.write(person[x] + " ");
}
</script>
</body>
</html
```

# JavaScript
## Estruturas de Controle (break e continue)

- Em todas as estruturas de controle é possível utilizar os comandos: break e continue
- break
  - Finaliza a execução do laço independente da condição
- Continue
  - Encerra a execução de um laço e continua no próximo passo
- for

# Javascript
# Exemplos - Eventos

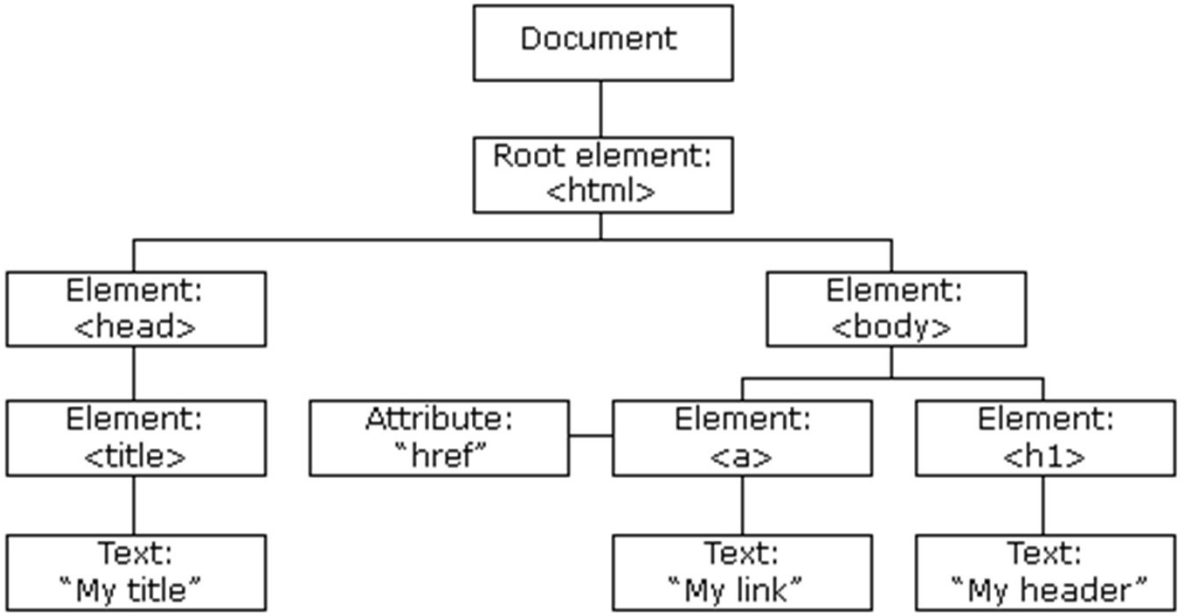
```
<html>
<head>
<script type="text/javascript">
function displayDate(){
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
<h1>My First Web Page</h1>
<p id="demo">This is a paragraph.</p>
<button type="button" onclick="displayDate()">Display
Date</button>
</body>
</html>
```

# Javascript
# Exemplos - Validação

```
<html>
<head>
<script type="text/javascript">
function validateForm(){
var x=document.forms["myForm"]["email"].value
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<1 || dotpos<=atpos+2 || dotpos+2>=x.length) {
  alert("Not a valid e-mail address");
  return false;
  }
}
</script>
</head>
<body>
<form name="myForm" action="x" onsubmit="return validateForm();" method="post">
Email: <input type="text" name="email">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

# JavaScript Document Object Model (DOM)

- Quando uma página é carregada o navegador cria um modelo hierárquico do documento
- O objeto “document” permite o acesso a todo o conteúdo HMTL
  - `document.getElementById("demo").innerHTML = "Hello World!";`
- Maiores informações
  - https://www.w3schools.com/js/js_htmldom_document.asp



# JavaScript
# Browser Object Model (BOM)

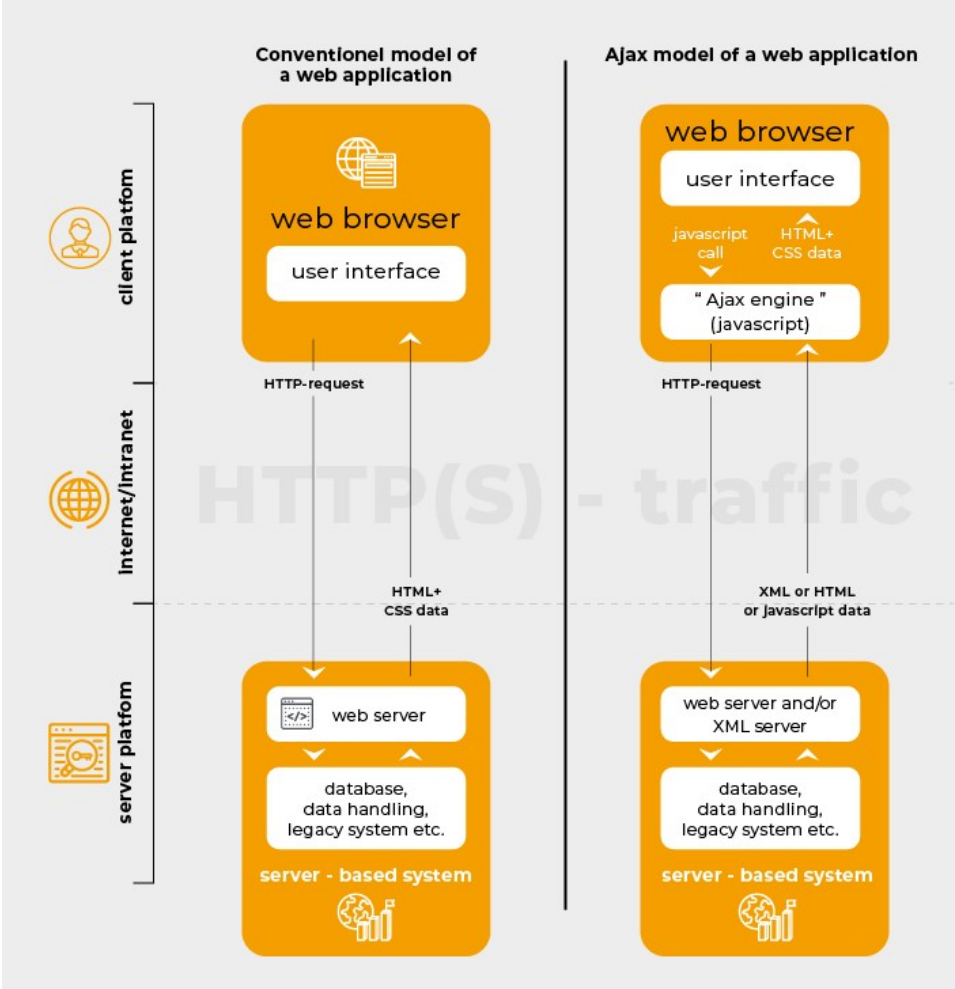
- O navegor também disponibiliza um objeto que representa o navegador e a janela que o contém
- O Brower Object Model (BOM) permite o acesso a diversas propriedades
  - `window.document.getElementById("header");`
- Propriedades englobam: janela; histórico; localização
- https://www.w3schools.com/js/js_window.asp

# Asynchronous JavaScript and XML (AJAX)

- É um conceito de programação do lado cliente que permite criar aplicações web com chamada assíncronas
- Utilizando o AJAX aplicações web podem obter dados de forma assíncrona do servidor web e alterar os dados exibidos na página HTML
- Assim o conteúdo da página web é alterado de forma dinâmica sem a necessidade de recarregar toda a página
- Geralmente o AJAX utiliza JSON ou XML para exibir os dados que são obtidos
- O AJAX utiliza as seguintes tecnologias
  - HMTL e CSS para apresentação
  - Document Object Model (DOM) para interação com elementos da página
  - JSON ou XML para troca de dados
  - XMLHttpRequest para a comunicação assíncrona
  - Javascript para realização da chamada e lógica de controle

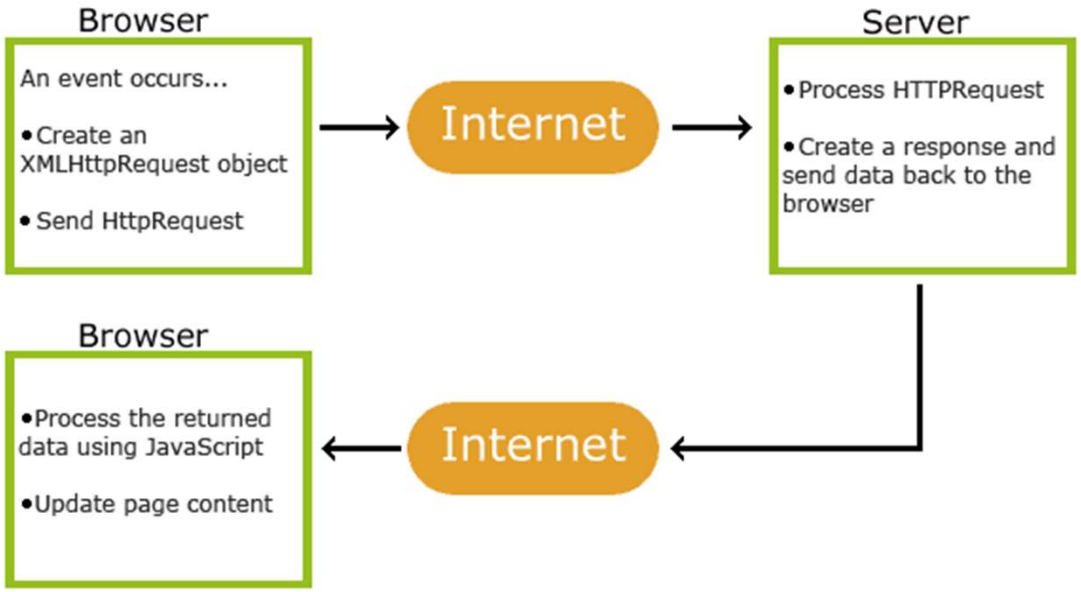
# Asynchronous JavaScript and XML (AJAX)

- Aplicação We Convencional x Aplicação Web baseada em AJAX
- Com o AJAX é possível
  - Obter dados do servidor após a página HTML ser carregada no navegador
  - Atualizar uma página web sem recarregar todo o seu conteúdo
  - Enviar dados para o servidor web em segundo plano
- Em JavaScript o objeto responsável por este tipo de chamada é o XMLHttpRequest



# Como o Ajax Funciona

1. Um evento ocorre em uma página da web. Por exemplo, a página é carregada ou há um clique em um botão é clicado
2. Um objeto XMLHttpRequest é criado pelo motor de JavaScript no navegador
3. O objeto XMLHttpRequest envia uma solicitação a um servidor web
4. O servidor processa a solicitação
5. O servidor envia uma resposta de volta à página da web
6. A resposta é lida pelo JavaScript
7. Ação adequada é realizada por JavaScript e no geral envolve uma alteração no conteúdo HTML da página



# Exemplo de uso

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
    this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>
</body>
</html>
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_first

# AJAX em Javascript

- Criar o objeto

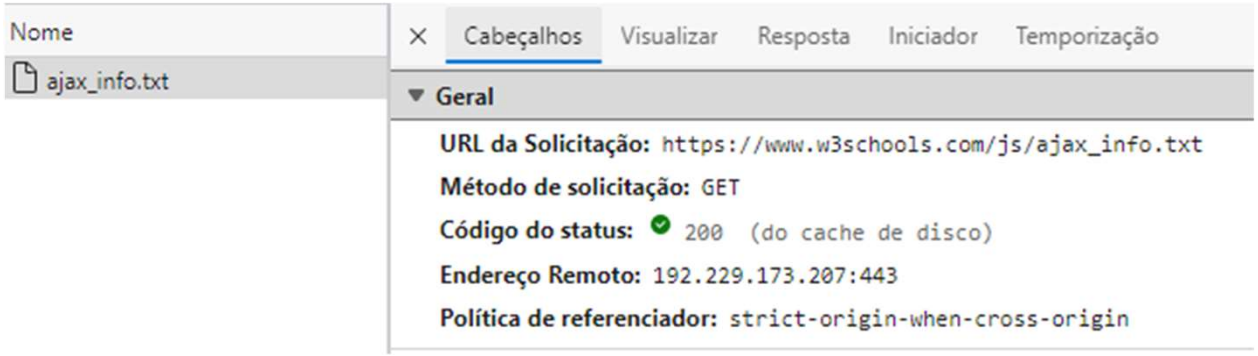
```
obj = new XMLHttpRequest();
```

- Definir a função de callback

```
xhttp.onload = function() {
  // What do do when the response is ready
}
```

- Enviar o pedido

```
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```



# Exemplo II – Sugestões
# Serviço em PHP no backend

```
<!DOCTYPE html>
<html>
<body>
<h2>The XMLHttpRequest Object</h2>
<h3>Start typing a name in the input field below:</h3>
<p>Suggestions: <span id="txtHint"></span></p>
<p>First name: <input type="text" id="txt1" onkeyup="showHint(this.value)"></p>
<script>
function showHint(str) {
  if (str.length == 0) {
    document.getElementById("txtHint").innerHTML = "";
    return;
  }
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("txtHint").innerHTML =
    this.responseText;
  }
  xhttp.open("GET", "gethint.php?q="+str);
  xhttp.send();
}
</script>
</body>
</html>
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_suggest_php

# Exemplo II

```
<!DOCTYPE html>
<html>
<style>
th,td {
  padding: 5px;
}
</style>
<body>
<h2>The XMLHttpRequest Object</h2>
<form action="">
  <select name="customers"
onchange="showCustomer(this.value)">
    <option value="">Select a customer:</option>
    <option value="ALFKI">Alfreds Futterkiste</option>
    <option value="NORTS ">North/South</option>
    <option value="WOLZA">Wolski Zajazd</option>
  </select>
</form>
<br>
<div id="txtHint">Customer info will be listed here...</div>
```

```
<script>
function showCustomer(str) {
  if (str == "") {
    document.getElementById("txtHint").innerHTML =
"";
    return;
  }
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("txtHint").innerHTML =
this.responseText;
  }
  xhttp.open("GET", "getcustomer.php?q="+str);
  xhttp.send();
}
</script>
</body>
</html>
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_database

# JavaScript Web APIs

- JavaScript Validation API
- Web History API
- Web Storage API
- Web Workers API
- Fetch API
- Web Geolocation API

# JavaScript Web APIs

- JavaScript Validation API
  - Validação de dados em formulários
  - https://www.w3schools.com/js/js_validation_api.asp
- Web History API

- Web Storage API
- Web Workers API
- Fetch API
- Web Geolocation API