Programação Visual e Autoria Web

Introdução a JavaScript

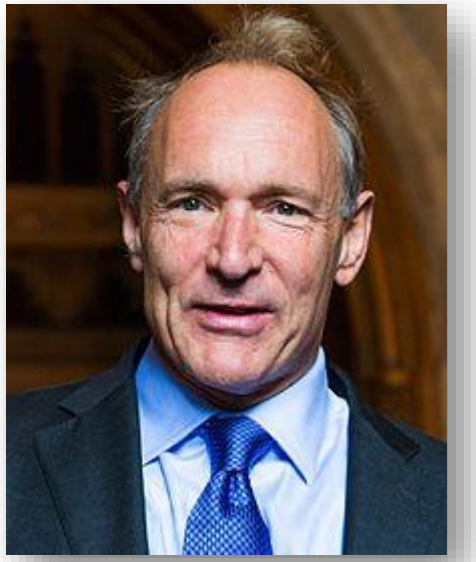
Universidade Federal do Rio Grande do Norte

Introdução

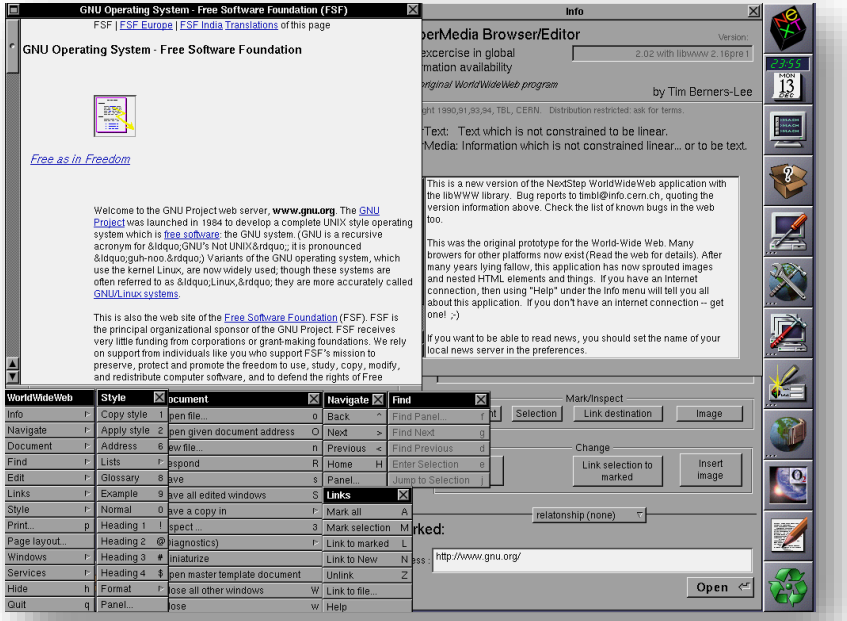
- Javascript é uma das **linguagens de programação mais populares do mundo**.
  - Linguagem mais utilizada em repositórios no GitHub
- Seu uso mais comum é como uma linguagem **de scripting do lado do cliente**, mas hoje em dia o uso de Javascript é muito mais vasto.
  - É possível criar soluções para web apenas com Javascript
- Quando escrevemos páginas **apenas com HTML** temos uma **página estática** que não reage a interação com o cliente.

Breve histórico

- Em 25 de dezembro de 1990 foi lançado o primeiro browser.
- Chamava-se WorldWideWeb (depois chamado de Nexus) e foi criado por Tim Berners-Lee.
  - Também desenvolveu o primeiro servidor web (CERN HTTPd).
- Esses eram os primeiros anos da internet e ela não era popular.



FONTE: https://en.wikipedia.org/wiki/Tim_Berners-Lee



FONTE: https://en.wikipedia.org/wiki/WorldWideWeb

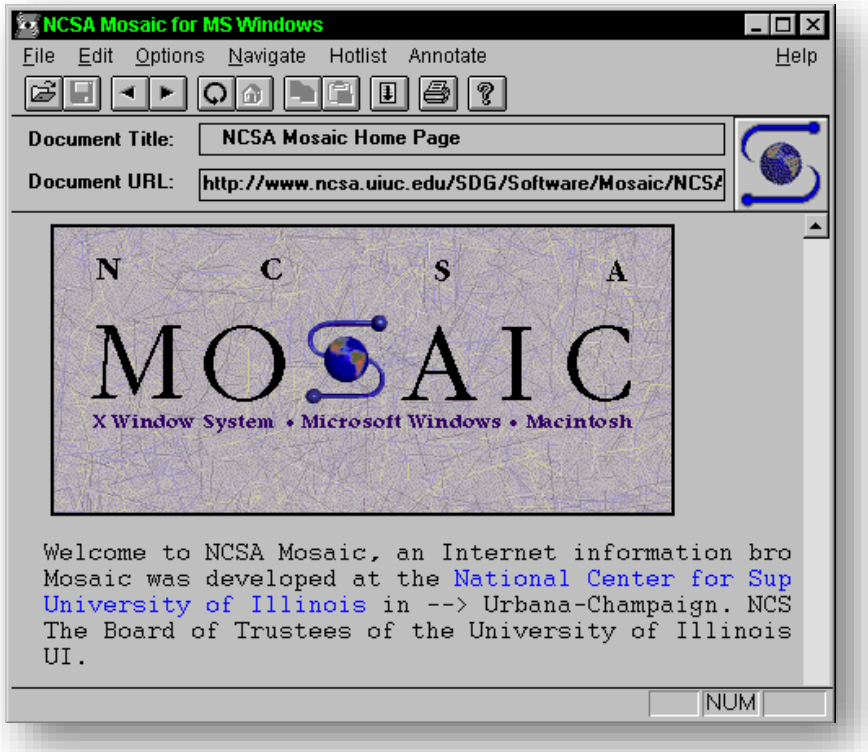
Breve histórico

- Em 1991 é promulgado nos EUA o **High Performance Computing Act.**
- Levou ao desenvolvimento da **infraestrutura nacional de informação** e ao financiamento **da Rede Nacional de Pesquisa e Educação** (NREN).
- **Estimulou desenvolvimentos tecnológicos significativos**, como o browser Mosaic e a criação de uma rede de computadores de fibra óptica de alta velocidade.

FONTE: https://en.wikipedia.org/wiki/High_Performance_Computing_Act_of_1991

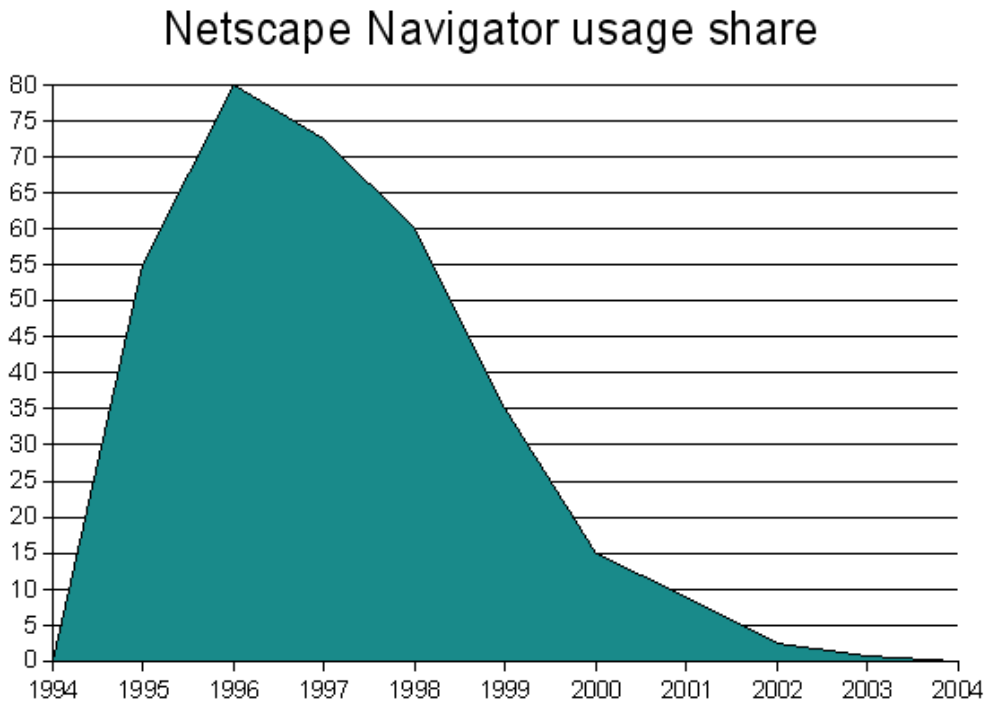
Breve histórico

- Mosaic foi desenvolvido por **Marc Andreessen and Eric Bina** na NCSA (National Center for Supercomputing Applications) que opera como unidade da universidade de Illinois (EUA).
- Lançado para sistemas UNIX em 1993 e mais tarde portado para Macintosh e Windows, tornando-se um browser bastante popular.
- Até esse momento não existe o Javascript.
  - O conceito de DOM (Document Object Model) já existe mas não está nem perto de ser padronizado.



Breve histórico

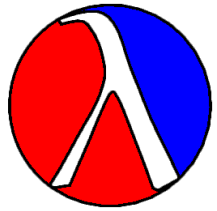
- Em 1993 Marc Andreessen se forma e se muda para a California para co-fundar a empresa responsável por desenvolver o browser **Netscape Navigator**.

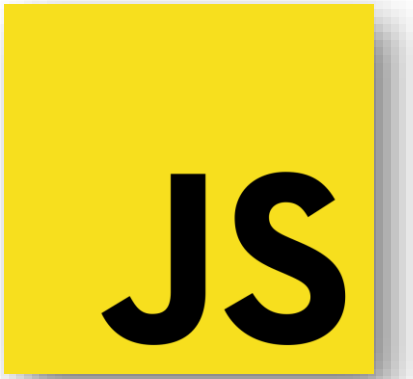


FONTE: https://commons.wikimedia.org/wiki/File:Netscape_Navigator_usage_share.png

Breve histórico

- Os desenvolvedores do Netscape perceberam que era preciso dar mais **dinamicidade** para o cliente web.
- Em abril de 1995 **Brendan Eich** entra para a equipe de desenvolvimento do Netscape com o objetivo de adicionar a linguagem de programação **Scheme a um browser mas mantendo uma sintaxe que parecesse com Java**.
- A primeira versão foi concluída em **dez dias** para acomodar o cronograma de lançamento do Navigator 2.0 Beta
  - Chamada de Mocha
  - Renomeada para LiveScript em setembro de 1995
  - Posteriormente renomeada para JavaScript no mesmo mês.





Breve histórico

- Nesse mesmo período, agosto de 1995, a Microsoft lança o browser Internet Explorer.
  - Na versão 3.0 o Internet Explorer utilizava o Jscript para tornar páginas dinâmicas.
  - Segundo a Wikipédia, o Jscript era baseado no Javascript na Netscape.
- Jscript e Javascript começam a crescer e surge então necessidade de padronização das linguagens para browser.
- **European Computer Manufacturers Association** (ECMA), organização fundada em 1961 para padronizar sistemas de computadores na Europa fica a cargo da padronização do Javascript.
  - Junho de 1997 é divulgado o padrão ECMA-262, ou ECMAScript

Breve histórico

- O padrão ECMA 262 oferece aos desenvolvedores:
  - Especificação consistente
  - Orientação para implementação do Javascript
- Muito similar ao Javascript moderno mas sem algumas características como:
  - Blocos try catch
  - Expressões regulares
  - Operador de igualdade estrita
- Em 1999 é lançava a versão 3 do EMCAScript, uma nova versão só seria lançada 10 anos depois.

Breve histórico

- Em 1999 a Netscape é comparada pela America Online.
- Internet Explorer já possui 80% do mercado de browsers.
  - Microsoft muitas vezes não seguia a regras da especificação.
  - Microsoft implementa suas próprias extensões para Javascript.
  - Algumas são problemas até hoje quando se quer dar suporte a browsers muito antigos.
  - Outras são caso de sucesso como a Ajax que torna possível executar código assíncrono, precursor das single-page applications.

Breve histórico

- Em 2000 estavam em andamento os trabalhos para especificação do EMCAScript 4.
  - O código parecia com algo com o TypeScript moderno.
  - Tipos, classes, interfaces, etc.
  - Foco era a escala empresarial.
- Criou-se um empasse entre a versão 3.1 (que trazia apenas pequenas correções) e a versão 4.0 que mudava a linguagem completamente.
  - 2008 a versão 4.0 é abandonada (chega ao mercado como ActionScript da Macromedia para o Flash)

Breve histórico

- Em 2006 surge A JQuery uma biblioteca de funções JavaScript que interage com o HTML, **desenvolvida para simplificar os scripts interpretados no navegador do cliente**.
  - É a mais popular das bibliotecas JavaScript.
- Em 2008 surge o V8, um mecanismo JavaScript de código aberto desenvolvido pelo The Chromium Project para navegadores da web Google Chrome e Chromium.
  - V8 compila o JavaScript diretamente no código da máquina nativo antes de executá-lo.
  - O código compilado é adicionalmente otimizado (e re-otimizado) dinamicamente no tempo de execução.

Breve histórico

- Em maio de 2009 **Ryan Dahl** apresenta o Node.js um ambiente de execução de JavaScript de código aberto executa o código JavaScript fora de um navegador.
  - Construído sobre o V8.
  - O Node.js representa o paradigma de "JavaScript em todos os lugares“
- Em dezembro de 2009 é lançada a ECMAScript 5, que basicamente era o ECMAScript 3.1

Breve histórico

- ES5 possui características muito importantes:
  - Suporte a JSON
  - Métodos de Array
  - Modo estrito
- Começam então a surgir Frameworks JS para single-page applications como Angular e Backbone (ambos em outubro de 2010)

Breve histórico

- Em 2009 um programador chamado Jeremy Ashkenas, co-criador do Backbone.js, cria a linguagem CoffeeScript que compila para JavaScript, um precursor dos **transpiladores**.
- Transpiladores são softwares que convertem o código de uma versão da ECMAScript em outra, por exemplo, de ES6 para ES5.
  - Essa característica é extremamente importante para manter a compatibilidade de versões.
  - Um dos transpiladores mais conhecidos é o Babel.js.

Breve histórico

- Novidades ES6
  - Promisses
  - let e const
  - Arrow Functions
  - Classes
- Pela transpilação desenvolvedores podem utilizar características modernas e converter seu código para uma versão anterior do JavaScript.

Breve histórico

- 6th Edition - ECMAScript 2015
- 7th Edition - ECMAScript 2016
- 8th Edition - ECMAScript 2017
- 9th Edition - ECMAScript 2018
- 10th Edition - ECMAScript 2019
- 11th Edition - ECMAScript 2020
- 12th Edition - ECMAScript 2021
- 13th Edition - ECMAScript 2022

Principais características

- Interpretada
- Independente de plataforma
- Imperativa e Estruturada
- Baseada em objetos
- Tipagem dinâmica
- Fracamente tipada
- Avaliação em tempo de execução

Como usar JavaScript numa página web

- Podemos adicionar um código JavaScript em uma página de duas formas:
- Utilizando o elemento <script> </script> e colocando o código JS como conteúdo do elemento.
- Criando um arquivo com extensão .js adicionado através do atributo `src` do elemento <script>

Como usar JavaScript numa página web

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <meta http-equiv="X-UA-Compatible" content="ie=edge" />
        <title>Aula 10</title>
        <script>
            alert("Hello World!");
        </script>
    </head>
    <body></body>
</html>
```

Como usar JavaScript numa página web

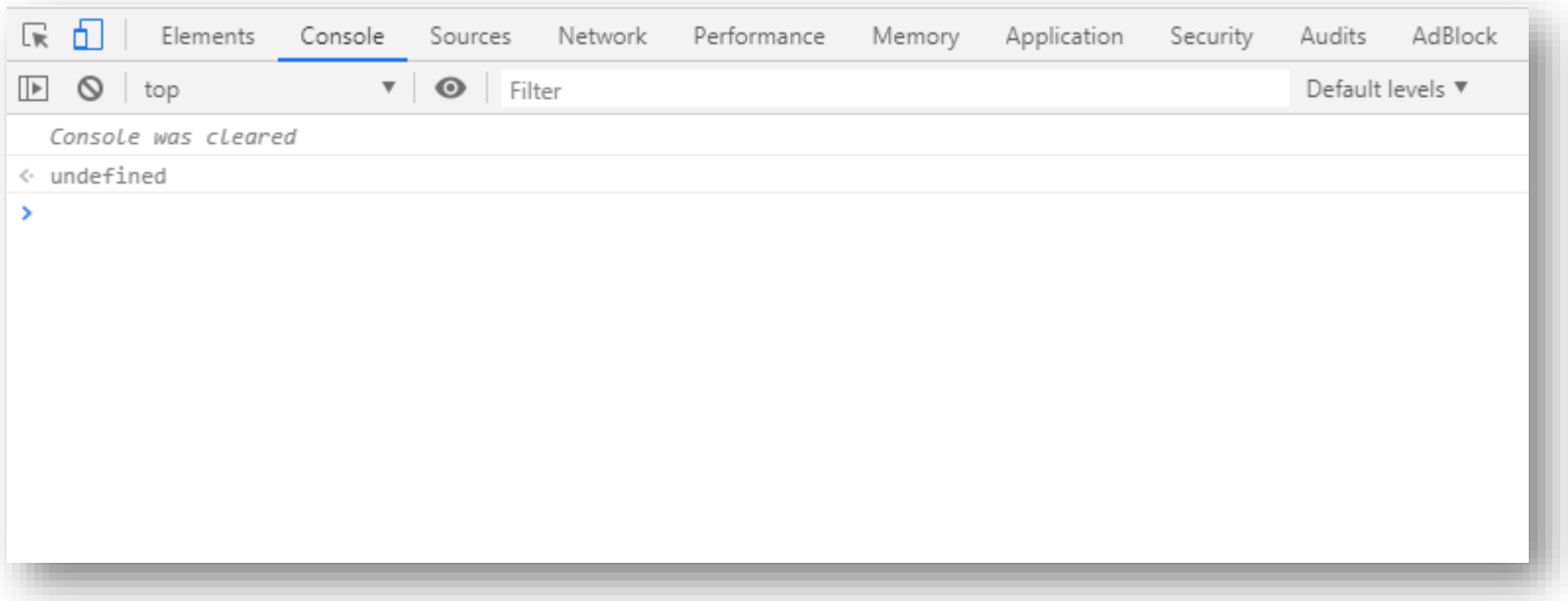
```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <meta http-equiv="X-UA-Compatible" content="ie=edge" />
        <title>Aula 10</title>
        <script src="script.js"></script>
    </head>
    <body></body>
</html>
```

script.js

```js
alert("Hello World!");
```

Console do browser

- O console do browser será uma ferramenta para analisar e executar código JavaScript.
- No browser Chrome, aperte f12 e selecione a aba “Console”.



Tipos de dados

- O JavaScript possui os seguintes tipos de dados:
  - Número
  - Literal
  - Booleano
  - Função
  - Objeto
- Podemos saber o tipo de uma variável com o uso do operador `typeof <variavel>`

```
let variavel = 5
typeof variavel
number
```

Tipos de dados

```
let num = 3;
num = "três";
let preco = 1.99;
let nome = "tads";
let verdadeiro = true;
let nulo = null;
let naoDefinido;

console.log(num);
console.log(preco);
console.log(nome);
console.log(verdadeiro);
console.log(nulo);
console.log(naoDefinido);
```

```
const constante = 3;
constante = 5;
```

- **`let`** é usado para declarar variáveis
- **`const`** é usado para declarar variáveis finais (constantes)
- Versões antigas do Javascript usam a palavra **`var`**.
  - var ou let ?

Tipos de dados

```
let num = 3;
num = "três";
let preco = 1.99;
let nome = "tads";
let verdadeiro = true;
let nulo = null;
let naoDefinido;

document.writeln(num);
document.writeln(preco);
document.writeln(nome);
document.writeln(verdadeiro);
document.writeln(nulo);
document.writeln(naoDefinido);
```

```
const constante = 3;
constante = 5;
```

- **`let`** é usado para declarar variáveis
- **`const`** é usado para declarar variáveis finais (constantes)
- Versões antigas do Javascript usam a palavra **`var`**.
  - var ou let ?

Tipos de dados

```
let numeros = [1, 2, 3];
let p = { nome: "Taniro", idade: 33 };

console.log(p.nome);
console.log(numeros);
console.log(numeros[0]);
console.log(typeof p);
console.log(typeof numeros);

let calcula = function(a) {
    return a * 2;
};

console.log(calcula(5));
console.log(typeof calcula);
```

```
const constante = 3;
constante = 5;
```

- **`let`** é usado para declarar variáveis
- **`const`** é usado para declarar variáveis finais (constantes)
- Versões antigas do Javascript usam a palavra **`var`**.
  - var ou let ?

Operadores

Operador aritmético	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão (*)
++	Incremento
--	Decremento

Operadores

Operador de atribuição	Descrição
=	Atribuição
+=	Atribuição da soma
-=	Atribuição da subtração
*=	Atribuição da multiplicação
/=	Atribuição da divisão
%=	Atribuição de resto

Operadores

Operador de comparação	Descrição
==	Igual a
===	Igual a (tanto tipo quanto valor)
!=	Diferente de
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a

Operadores

Operador lógico	Descrição
&&	E
||	OU
!	Negação

Operadores

Operador bit a bit	Descrição
&	E
|	Ou
~	Negação
^	Ou exclusivo
<<	Deslocamento à esquerda
>>	Deslocamento à direita

Truthy e Falsy

- Em JavaScript variáveis podem assumir o valor de verdadeiro ou falso de maneira um pouco diferente das outras linguagens de programação.
  - Reflexo da tipagem dinâmica e fraca

```
if ("bolinha"){
    console.log("Verdadeiro")
}else{
    console.log("Falso")
}
```

  - A saída no console será “Verdadeiro”

Truthy e Falsy

Tipo do valor	Resultado
Undefined	False
Null	False
Booleano	Verdadeiro se `true`, Falso se `false`.
Número	Falso para +0, -0 ou NAN; caso contrário Verdadeiro
Literal	Falso se o tamanho da String é zero; caso contrário Verdadeiro
Objeto	Verdadeiro

Operadores == e ===

- O operador == pode considerar dois valores iguais mesmo se eles forem de tipos diferentes.
- O operador === é mais simples de compreender, pois ele funciona como nas outras linguagens de programação, ou seja, um valor só é igual a outro se possuem mesmo valor e mesmo tipo.

Estruturas de controle

- JavaScript possui um conjunto de estruturas de controle semelhante ao das linguagens C e Java.
- Condicionais:
  - `if ... else`
  - `switch`
- Laços:
  - `while`
  - `do ... while`
  - `for`

if ... else

```
if ("bolinha") {
    console.log("Verdadeiro");
} else {
    console.log("Falso");
}
```

```
numero === 5 ? numero++ : numero--;
```

```
if ("bolinha") {
    console.log("Verdadeiro");
} else if (mes == 8){
    console.log("Falso");
}
```

switch

```
switch (mes) {
    case "janeiro":
        console.log(1);
        break;
    case "fevereiro":
        console.log(2);
        break;
    case "marco":
        console.log(3);
        break;
    case "abril":
        console.log(4);
        break;
    default:
        console.log("Não encontrado");
}
```

Laços

```
for (let i = 0; i < 10; i++) {
    console.log(i);
}
```

```
let i = 0;
do {
    console.log(i++);
} while (i < 10);
```

```
let i = 0;
while (i < 10) {
    console.log(i++);
}
```

Referências

- The Weird History of JavaScript. Disponível em: https://www.youtube.com/watch?v=Sh6lK57Cuk4. Acesso em 08/03/2021.

Bibliografia Básica

- DUCKETT, J. **Javascript e Jquery**: desenvolvimento de interfaces web interativas. Rio de Janeiro: Alta Bokks, 2016
- FLANAGAN, D. **JavaScript**: O Guia Definitivo. Bookman, 2012. ROGERS, Y.; SHARP, H.; PREECE, J. **Design de Interação: além da interação homem-computador**. Bookman, 2013. MEYER
- Web Content Accessibility Guidelines (WCAG) 2.1. Disponível em: <https://www.w3.org/TR/WCAG21/>
- https://developer.mozilla.org/en-US/docs/Web/HTML/
- https://www.w3schools.com/tags/
- https://css-tricks.com/archives/

Bibliografia Complementar

- AMARAL, L. G. **CSS Cascading Style Sheets**: guia de consulta rápida. 2. ed. São Paulo: Novatec, c2006.
- KAWANO, W. **Crie aplicativos Web com HTML, CSS, JavaScript, PHP, PostgreSQL, Bootstrap, AngularJS e Laravel**. Rio de Janeiro: Ciência Moderna, 2016.
- PUREWAL, S. **Aprendendo a Desenvolver Aplicações Web**. Novatec, 2014.
- TONSIG, S. L. **Aplicações na nuvem**: como construir com HTML5, javaScript, CSS, PHP e MYSQL. Rio de Janeiro: Ciência moderna, 2012.
- USABILIDADE.com. Disponível em <http://www.usabilidade.com/>
- TASK-Centered User Interface Design: A Practical Introduction. Disponível em <http://hcibib.org/tcuid/>