LINGUAGEM DE PROGRAMAÇÃO JAVASCRIPT

Felipe Augusto
Jonatas Fontele
Magno de Sousa

Sumário

- Histórico;
- Utilização;
- Características Estudadas;
- Vantagens;
- Desvantagens.

Histórico

- JavaScript foi originalmente desenvolvido em 1995 por Brendan Eich quando trabalhou na Netscape sob o nome de Mocha, posteriormente teve seu nome mudado para LiveScript e por fim JavaScript.

- JavaScript ≠ Java: Jogo de Marketing?

- Em 1996 a Microsoft criou o Jscript, uma linguagem compatível e similar a JavaScript, ocasionando na normatização do JavaScript.

- O nome JavaScript já era patenteado pela Sun Microsystems e não poderia ser usado. Portanto, o nome usado foi ECMAScript. Mesmo com esse nome, até hoje a linguagem é conhecida por JavaScript. ECMAScript só é usado para se referir às versões da linguagem.

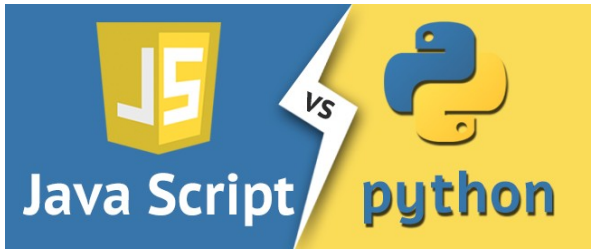
Histórico

Nov 2017	Nov 2016	Change	Programming Language	Ratings	Change
1	1		Java	13.231%	-5.52%
2	2		C	9.293%	+0.09%
3	3		C++	5.343%	-0.07%
4	5	^	Python	4.482%	+0.91%
5	4	v	C#	3.012%	-0.65%
6	8	^	JavaScript	2.972%	+0.27%
7	6	v	Visual Basic .NET	2.909%	-0.26%
8	7	v	PHP	1.897%	-1.23%
9	16	^^	Delphi/Object Pascal	1.744%	-0.21%
10	9	v	Assembly language	1.722%	-0.72%
11	19	^^	R	1.605%	-0.11%

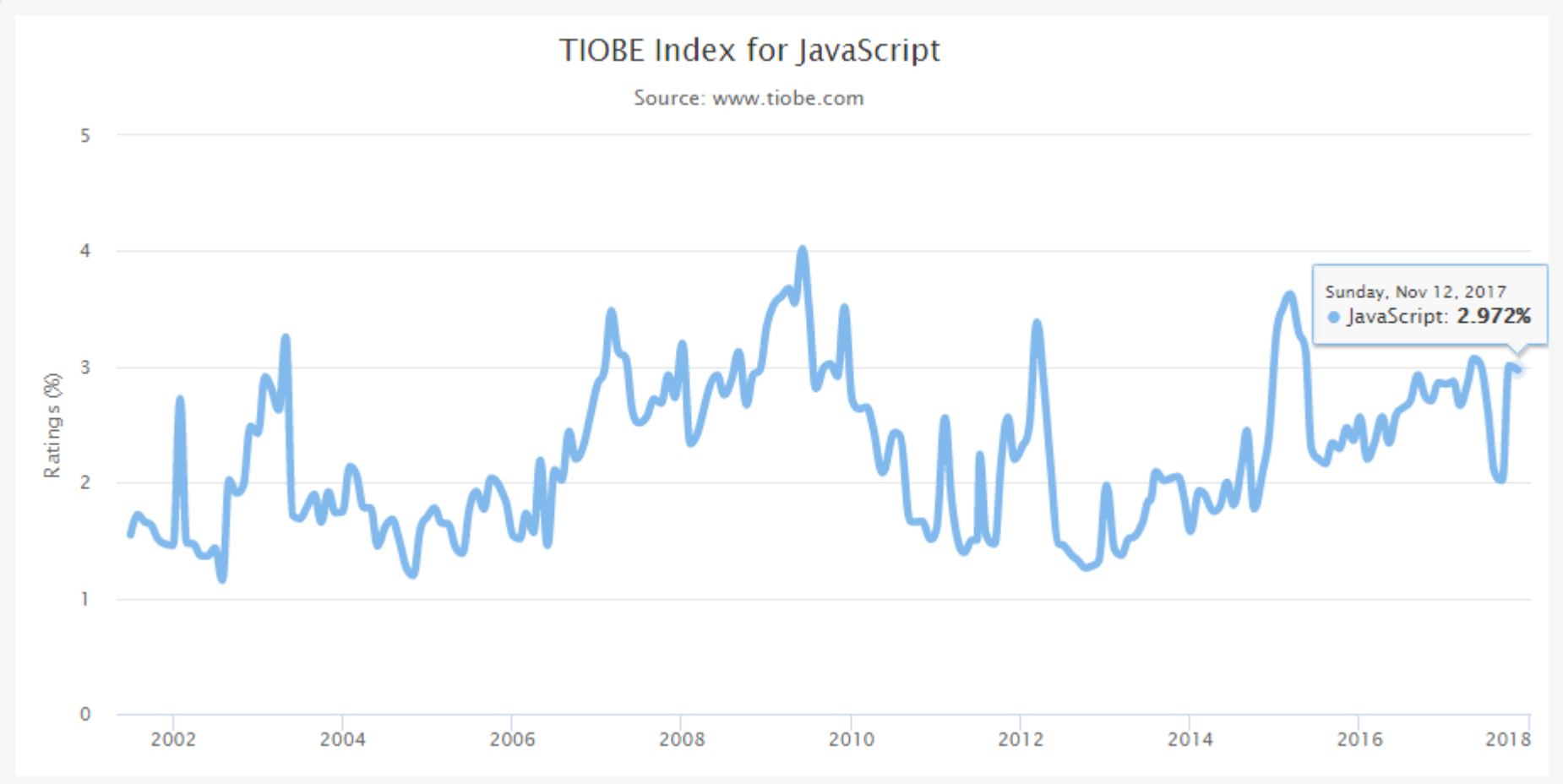
Histórico

- Posição mais alta (desde 2001): #6 em novembro de 2017;
- Posição mais baixa (desde 2001): #12 em outubro de 2014;
- Linguagem do ano: 2014.

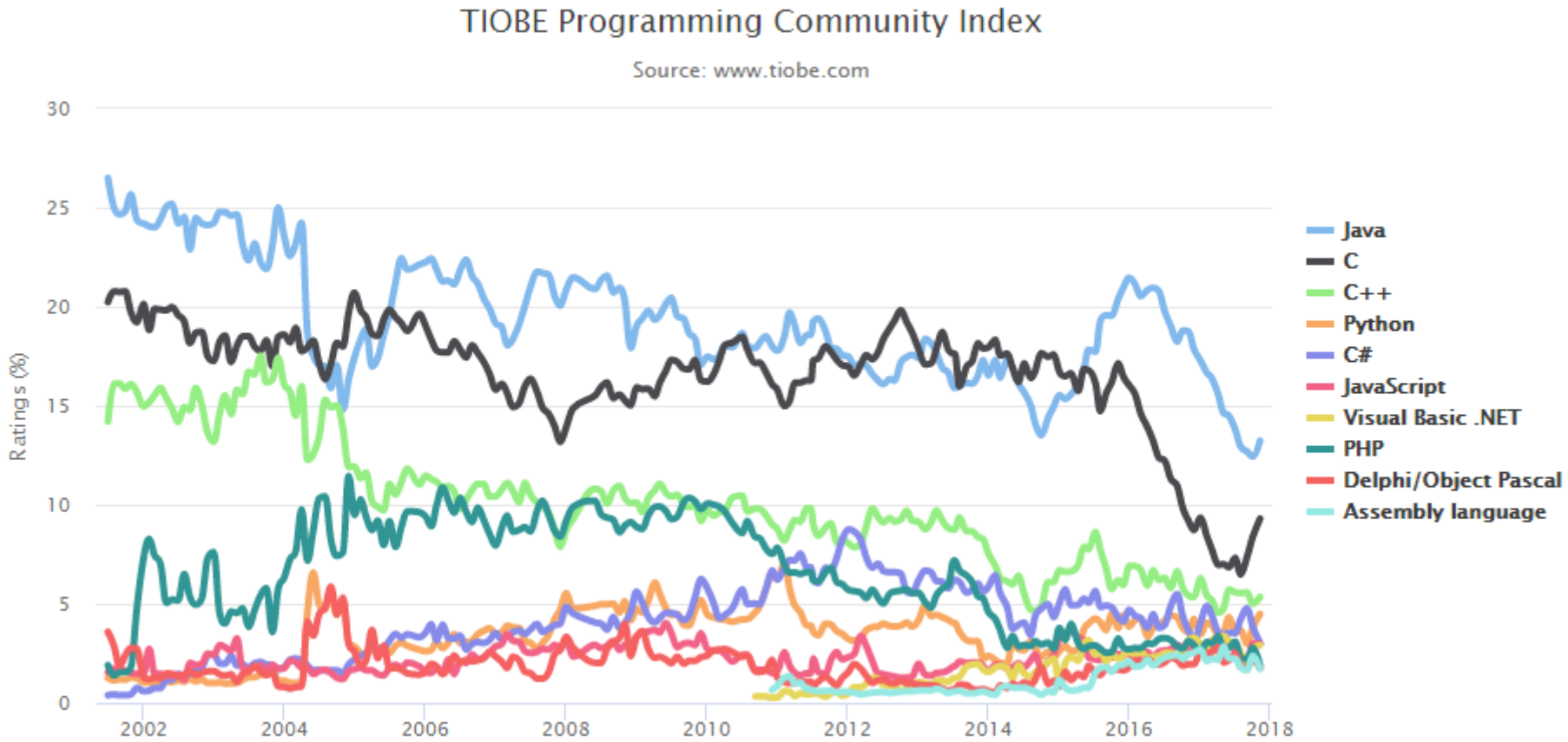
- Houve um tempo em que linguagens de script eram o futuro. Fácil de escrever, fácil de executar. Hoje as linguagens de script estão indo gradualmente para fora do top 20. Apesar do JavaScript ter subido um pouco, somente Python é considerado forte.



Histórico


TIOBE Index for JavaScript
Source: www.tiobe.com
Ratings (%)
5
4
3
2
1
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
Sunday, Nov 12, 2017
JavaScript: 2.972%

Histórico


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
Java
C
C++
Python
C#
JavaScript
Visual Basic .NET
PHP
Delphi/Object Pascal
Assembly language

Utilização

- O JavaScript é utilizado para escrever funções que são incluídas em páginas HTML e que interagem com o Modelo de Objeto de Documentos (DOM) da página.

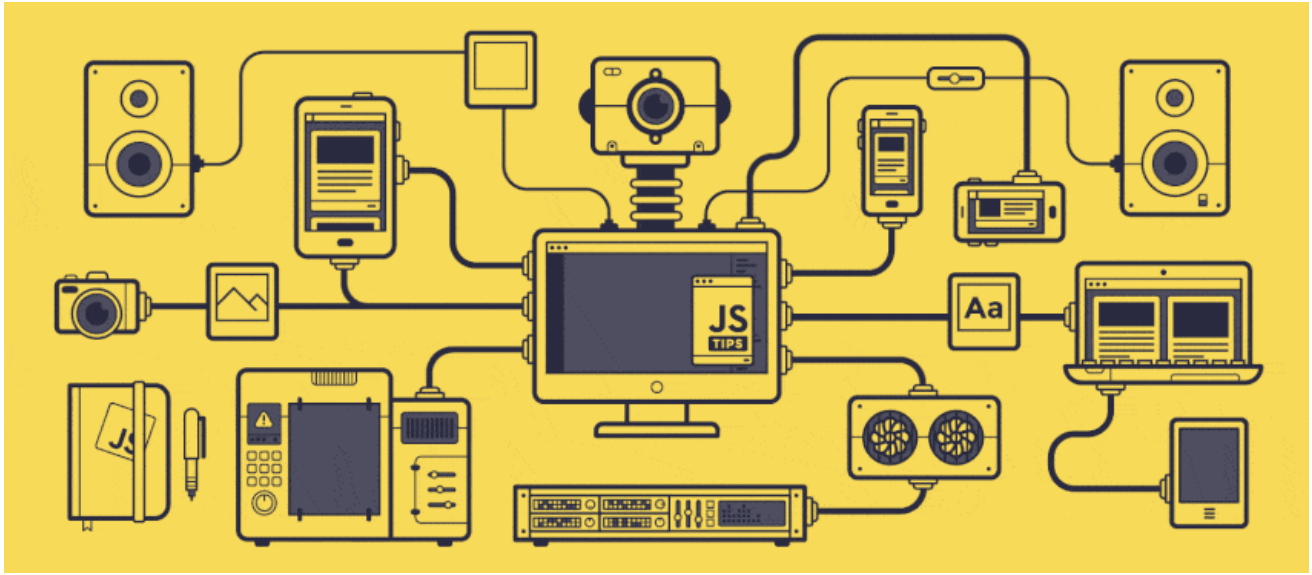
- Foi originalmente implementado como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor.

Principais Aplicações

- Interagir com o usuário, respondendo a eventos provocados por ele;
- Controlar o navegador
  - Carregar um novo documento;
  - Retornar para a página anteriormente visitada;
  - Abrir e fechar janelas;
  - Exibir páginas diferentes de acordo com o navegador do usuário;
- Controlar o conteúdo de formulários HTML;
- Manipular imagens embutidas no documento como nos menus do tipo pop-up;
- Interagir com camadas (layers);
- Ler e escrever o estado do cliente em Cookies;

Utilização Atual

- Animações parallax,
- Uma galeria de imagem;
- Interação com a rolagem da página,
- Validações e manipulação de elementos HTML e CSS.
- Até banco de dados NoSQL podem ser baseados em JavaScript.



Utilização Atual

- Validação de formulários: assim, o seu usuário não passa pela frustação de ter que recarregar a página inteira simplesmente porque digitou um campo errado;
- Esconder ou apresentar informações da página, conforme o comportamento de seu usuário;
- Janelas de aviso ou de confirmação;
- Sistemas de *autocomplete*, quando você digita na busca do Google e ele completa com as palavras possíveis ou já salvas pelo seu navegador;
- Sistemas ou aplicações web que mostram informações que mudam constantemente, conforme eventos internos ou externos ao site/aplicação.

Utilização Atual

facebook

Google

WIKIPEDIA
The Free Encyclopedia

twitter

YAHOO!

msn

Microsoft

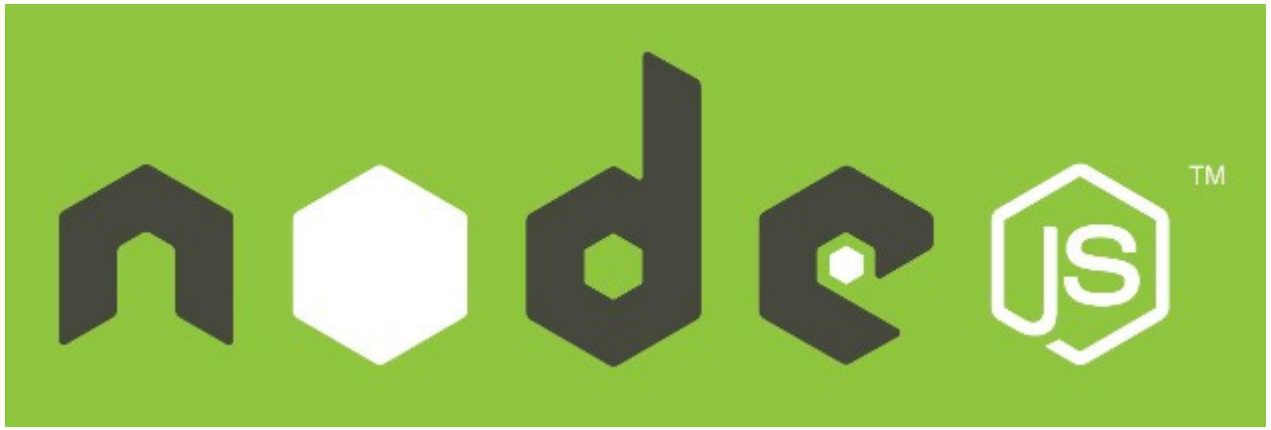
ebay

Pinterest

YouTube

amazon

Utilização Atual



Características

- **Facilidade de aprendizado?;**
- **Ortogonalidade;**
- **Alta:**
  - Reusabilidade;
  - Modificabilidade;
  - Portabilidade;
- **Baixa**:
  - Eficiência computacional para ganhar flexibilidade;
  - Legibilidade;
  - Redigibilidade;
  - Confiabilidade;

Características

- **Identificadores**
  - Não há limite para o número de caracteres;
  - São case sensitive;
  - A especificação JavaScript 1.5 define 89 palavras reservadas;
  - É permitido uso de caracteres especiais e caracteres de escape;
  - Usa escopo a nível de função;
  - A partir do JavaScript 1.7 passou a suportar escopo a nível de blocos através do comando *let*;

Características

- **Identificadores**
  - Escopo dinâmico.

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

    var numero1;
    var numero2;
    function soma(numero1,numero2){
        var resultado=numero1+numero2;
        function imprime(resultado){
            document.write(resultado);
        }
        function novoResultado(){
            resultado="<br>texto";
            // resultado=escape("<br>texto");
            imprime(resultado);
        }
    imprime(resultado);
    novoResultado();
    }


    soma(2,3);
</script>
```

Características

- **Identificadores**
  - Convenção de nomeação:
    - camelCase quando for nomear objetos, funções e instâncias;
    - PascalCase quando for nomear construtores;
    - underscore _ como primeiro caracter no nome de propriedades privadas;
    - Quando for guardar referência para this use _this.

Características

- **Tipos de dados**
  - Tipagem Dinâmica: como na maioria das linguagens de script, tipos são associados com valores, não com variáveis. Por exemplo, a variável x poderia ser associada a um número e mais tarde associada a uma string;
  - Number, como 7 ou 3,14, por exemplo;
  - Boolean;
  - String;
  - null;
  - undefined;
  - Object;
  - Vetores são trabalhados como Arrays.

Características

- **Variáveis**
  - Há 3 formas de declarar uma variável:
    - Com a palavra chave *var*. Por exemplo, var x = 42. Esta sintaxe pode ser usada para declarar tanto variáveis locais como variáveis globais.
    - Por simples adição de valor. Por exemplo, x = 42. Isso declara uma variável global. Essa declaração gera um aviso de advertência no JavaScript. Não é aconselhado usar essa variante.
    - Com a palavra chave *let*. Por exemplo, let y = 13. Essa sintaxe pode ser usada para declarar uma variável local de escopo de bloco.

Características

- **Constantes**
  - Pode criar uma constante apenas de leitura por meio da palavra-chave *const*. A sintaxe de um identificador de uma constante é semelhante ao identificador de uma variável: deve começar com uma letra, underline ou cifrão e pode conter caractere alfabético, numérico ou underline. Ex: const prefix = '212';
  - Uma constante não pode alterar seu valor por meio de uma atribuição ou ao ser declarada novamente enquanto o script é executado. Deve ser inicializada com um valor.
  - As regras de escopo para as constantes são as mesmas para as váriaveis *let* de escopo de bloco. Se a palavra-chave *const* for omitida, o identificador é adotado para representar uma variável.
  - Você não pode declarar uma constante com o mesmo nome de uma função ou variável que estão no mesmo escopo.

Características

- **Expressões**
  - Aritméticas: resulta em um número, por exemplo 3,14159. (Geralmente usa operadores aritméticos.);
  - String: avalia em uma cadeia de caracteres, por exemplo, "Fred" ou "234". (Geralmente usa operadores String.);
  - Lógicas: resulta em verdadeira (true em inglês) ou falsa (false em inglês). (Frequentemente envolve operadores lógicos.);
  - Object: resulta como um objeto. (Operadores especiais.)

Características

- **Operadores**
  - Operadores de atribuição. Ex: x += y
  - Operadores de comparação. Ex: 3 == var1
  - Operadores aritméticos. Ex: 12 % 5 returns 2
  - Operadores Bitwise. Ex: a | b (OR)
  - Operadores lógicos. Ex: expr1 || expr2 (Logical OR)
  - Operadores de sequência(String). Ex: "my " + "string"
  - Operadores especiais. (Operador condicional, operador vírgula, delete, in, instanceof, new, this, typeof, void.)

Características

- **Comandos**
  - JavaScript fornece também suporte a funções, aliado às facilidades da tipagem dinâmica, o que torna a definição de métodos simples e prática. Para criar funções, utilizamos a palavra reservada *function*.
  - As funções podem receber parâmetros e retornar valores, mas o tipo de retorno e o tipo dos parâmetros não precisa ser previamente definido.
  - Sem parâmetro e sem retorno:

```
function exibirMensagem()
{
 alert(“Olá, seja bem vindo(a)!”);
}
```

Características

- Com parâmetro e com retorno:

```
function somar(A, B)
{
 return A + B;
}
```

- Assim como a maioria das linguagens de alto nível, JavaScript possui estruturas condicionais e de repetição para controle de fluxo.

Características

```
if(condição 1)
{
 //ação se condição 1 verdadeira
}
else if (condição 2)
{
 //ação se condição 2 verdadeira
}
else
{
 //ação se nenhuma das condições for verdadeira
}
```

Características

- Exemplo de Switch:

```
switch(variável)
{
case valor1:
//ações caso valor1
break;
case valor2:
//ações caso valor2
break;
case valor3:
//ações caso valor3
break;
default:
//ações caso nenhum dos valores
break
}
```

Características

- **Modularização**
  - Encapsulamento em JavaScript
    - Encapsulamento é um dos fundamentos da programação orientada a objetos tradicional. Considerando que não temos classes no JavaScript e se entendermos encapsulamento como uma forma de restringir acesso à informação, concluímos que a definição de escopo é o caminho para alcança-lo.

    - A maneira mais fácil de alcançar encapsulamento é utilizando uma função anônima invocada imediatamente após sua definição:

```
function () {
        var hideFromOutside = true;
 })();
```

Características

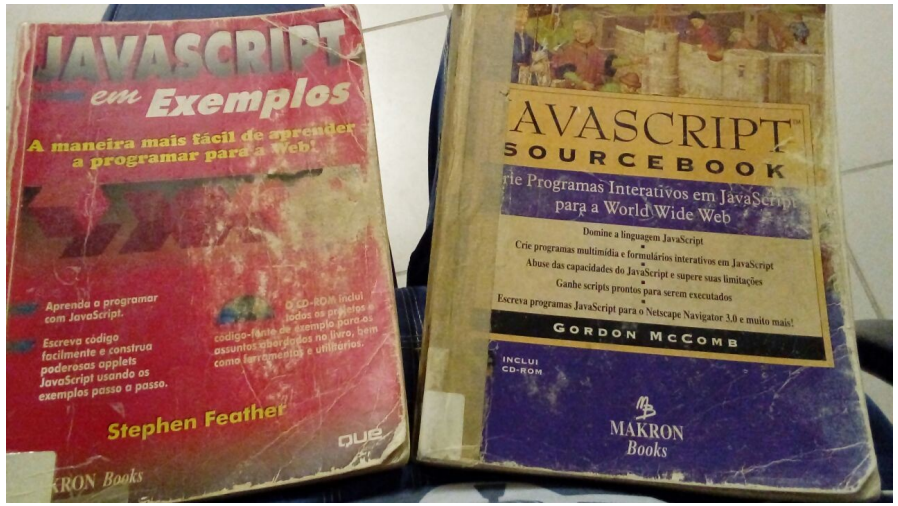
- **Paradigma**
  - **Imperativo: Estruturado e "Orientado" a Objetos**
    - Suporta os elementos de sintaxe de programação estruturada da linguagem C
    - É *dirigida* por eventos;
    - É quase inteiramente *baseada* em objetos. Objetos JavaScript são arrays associativos, aumentados com protótipos;
    - Usa protótipos em vez de classes para o mecanismo herança. É possível simular muitas características de orientação a objetos baseada em classes com protótipos;
    - Propriedades e seus valores podem ser adicionadas, mudadas, ou deletadas em tempo de execução;
    - A maioria das propriedades de um objeto (e aqueles em sua cadeia de herança via protótipo) pode ser enumerada usando-se uma estrutura de repetição for...in;
    - JavaScript possui um pequeno número de objetos padrão da linguagem como window e document.

Vantagens

- O código fonte é incluído no próprio arquivo HTML;
- Pelo fato do código JavaScript rodar localmente no navegador do usuário, e não em um servidor remoto, o navegador pode responder a ações mais rapidamente;
- É independente de plataforma;
- É uma linguagem segura;
- Pode detectar ações de usuário que o HTML sozinho não pode, tais como teclas pressionadas individualmente;
- Baseada em “orientação a objetos”;
- Sintaxe parecida com C, C++ e Java;
- Permite inserir vários efeitos, fazendo com que um site fique mais dinâmico;
- A linguagem é interpretada:
  - Facilidade para prototipação e depuração;
  - Flexibilidade na escrita do programa.

Desvantagens

- O código fonte é visível e poder ser lido por qualquer pessoa;
- Script tem de ser completamente baixado, assim, levará mais tempo do que se viesse de um servidor quando há um grande número de scripts;
- Não se pode afirmar se possui alta ou baixa legibilidade, redigibilidade ou confiabilidade pois vai depender qual assunto da linguagem estamos analisando;
- Baixa manutenibilidade;
- Difícil compatibilidade entre browsers;
- Maior consumo de espaço de memória;
- Execução mais lenta de um programa.



Bibliografia

- http://shipit.resultadosdigitais.com.br/blog/javascript-1-uma-breve-historia-da-linguagem/
- https://www.tiobe.com/tiobe-index/
- http://www.aminharadio.com/radio/html_aula21
- https://www.infoq.com/br/articles/mobile-architecture-html5-javascript
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/A_re-introduction_to_JavaScript
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Values,_variables,_and_literals
- https://developer.mozilla.org/pt-PT/docs/Web/JavaScript/Guia/Expressoes_e_Operadores
- http://www.nce.ufrj.br/ginape/js/conteudo/introducao/aplicacoes.htm
- https://becode.com.br/javascript-para-iniciantes-origens-o-que-e-para-que-serve/
- http://javascriptprogressivo.net/javascript-o-que-e-e-para-que-serve-tutorial/
- http://mariooliveira.info/aulas-programacao/javascript-aula03-identificadores.html
- https://msdn.microsoft.com/pt-br/library/2yfce773(v=vs.94).aspx