**UNIVERSIDADE COMUNITÁRIA DA REGIÃO DE CHAPECÓ - UNOCHAPECO**

WILLIAM KLASSMANN

**JAVASCRIPT MVC**

Criando um API´s REST com um framework JavaScript MVC

Chapecó

2015

WILLIAM KLASSMANN

**JAVASCRIPT MVC**

Criando um API´s REST com um framework JavaScript MVC

Trabalho de conclusão de curso, apresentado à UNOCHAPECO, como parte das exigências para a obtenção do título de Cientista da Computação.

Chapecó, 15 de Junho de 2014.

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Cezar Junior Souza

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. (Nome do professor avaliador)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. (Nome do professor avaliador)

SUMÁRIO

[**1.** **Introdução** 1](#_Toc413693954)

[**1.1** **Tema** 1](#_Toc413693955)

[**1.2** **Delimitação do problema** 1](#_Toc413693956)

[**1.3** **Questões de pesquisa** 1](#_Toc413693957)

[**2.** **Objetivos** 2](#_Toc413693958)

[**2.1** **Objetivo geral** 2](#_Toc413693959)

[**2.2** **Objetivo específico** 2](#_Toc413693960)

[**3.** **Justificativa** 3](#_Toc413693961)

[**4.** **Revisão bibliográfica** 4](#_Toc413693962)

[**5.** **Procedimentos metodológicos** 8](#_Toc413693963)

[**6.** **Cronograma** 9](#_Toc413693964)

[**7.** **Orçamento** 10](#_Toc413693965)

[**8.** **Referências** 11](#_Toc413693966)

1. **Introdução**

Segundo PLANKY (2014), aplicação REST (Transferência de Estado Representacional), é um design de arquitetura construído para servir aplicações em rede. Ao construir uma aplicação, todos os programadores planejam adquirir o máximo de qualidade do código, ter ótimo desempenho, possuir uma estrutura padronizada para que desta forma a manutenção do software seja mais simples e sem grandes complicações. Para garantir esses benefícios e auxiliar no desenvolvimento, programadores utilizam conjuntos de classes que provê a abstração para resolver uma família de problemas, conhecidos como Frameworks.

Este projeto irá levantar as vantagens e as desvantagens da utilização de um framework JavaScript MVC para criação de API´s REST, com a utilização do Angular.js, que é um framework que liga o HTML (views) com objetos JavaScript (models).

* 1. **Tema**

Criar uma API´REST utilizando um framework JavaScript MVC.

* 1. **Delimitação do problema**

Usar JavaScript é difícil sem a utilização de frameworks para auxiliar no desenvolvimento.

O código JavaScript geralmente é mal organizado em uma aplicação Web.

Falta de padronização no código JavaScript.

* 1. **Questões de pesquisa**

Por que utilizar um framework Java Script MVC?

O que seria a Model, View, Controller no Java Script MVC?

Como funciona e por que utilizar o Angular.js?

Qual o melhor framework para manipulação de DOM e de Objetos?

Quais são as diferenças entre Angular.js e jQuery?

Qual é o de mais fácil manutenção?

É possível utilizar vários Frameworks JavaScript ao mesmo tempo?

1. **Objetivos**
   1. **Objetivo geral**

Levantamento das vantagens e desvantagens de utilização de um framework JavaScript MVC para criação de API's Rest.

* 1. **Objetivo específico**

Identificar qual é o melhor framework para manipulação do DOM e de Objetos, para ter melhor desempenho.

Mostrar como funciona a manipulação do HTML e de objeto com o Angular.js.

Melhorar a estrutura, qualidade e o desempenho da aplicação.

Verificar qual dos dois frameworks é melhor para organização do código JavaScript.

Verificar o que é Model, View e Controller e como funciona o padrão MVC no JavaScript.

1. **Justificativa**

Para deixa os códigos organizados, grande maioria dos programadores utilizam ferramentas, frameworks, padrões de desenvolvimento, entre outros métodos.

Nas linguagens de desenvolvimento *server-side*, como o PHP, geralmente em grandes aplicações são utilizados frameworks, por exemplo o Zend Framework. Segundo CHASE (2006) existem vários frameworks, por exemplo, Zend Framework, Ruby on Rails, CakePHP que utilizam alguns conceitos novos da programação, que são:

* Orientação a objetos;
* Paradigma MVC (Model-View-Controller);

No JavaScript já é mais difícil ter uma organização de código, pois aplicações web estão utilizando cada vez mais JavaScript e muitas vezes não é enxergado como parte fundamental ou é ignorado pelos programadores. (SILVA, 2014).

Para organização de código JavaScript existem frameworks e também algumas práticas, com a ideia de padrões de desenvolvimento para facilitar a programação e a organização sem utilização de frameworks.

Ter um código bem organizado e padronizado traz vantagens muito importantes.

Algumas das vantagens são:

* Manutenção do código;
* Entendimento do código;
* Facilidade de manipular o código;

Para as empresas e programadores autônomos que possuem algum software ativo, a manutenção as vezes pode dar dor de cabeça, porque pode ser difícil encontrar algum erro ou acrescentar novos atributos ou regras na aplicação.

Já os programadores novos que chegam nas empresas para trabalharem, encontram muitas dificuldades em se adaptar na forma de programar. Se o código estiver bem organizado e padronizado, essa dificuldade diminui muito, e o novo programador acaba se adaptando facilmente e a empresa acaba ganhando tempo.

Por esses motivos, ter um código organizado, tanto server-side quanto *Client-side*, faz grande diferença.

Com essa dificuldade de organização de código JavaScript, falta de padronização, o objetivo do trabalho é criar um API REST utilizando o design patters MVC no JavaScript em conjunto com o framework Angular.js e analisar as suas vantagens e desvantagens em relação a utilização de JavaScript puro e outros frameworks, como o JQuery.

1. **Revisão bibliográfica**
   1. **Design patterns**

Hoje em dia, desenvolver um projeto não é fácil. Essa dificuldade começa quando não é definido claramente um problema. Existe uma carência muito grande na documentação e nas soluções encontradas dos problemas. (JUNQUEIRA; COSTA; LIRA, 1998).

O uso de *Design Patterns* (Padrão de projeto)vem para preencher esse buraco, tornando-se um mecanismo para compartilhamento de conhecimentos entre desenvolvedores. Dessa forma uma solução de um problema, pode ser novamente aplicada por outro desenvolvedor para suprir sua necessidade e facilitar seu trabalho. (JUNQUEIRA; COSTA; LIRA, 1998). “O foco principal não é nem tanto tecnológico, mas sim o de criar uma cultura de catalogação de experiências e soluções para apoiar o desenvolvimento de software.” (JUNQUEIRA; COSTA; LIRA, 1998).

* + 1. **Origens**

Os primeiros padrões de projetos surgiram por volta dos anos 80 da engenharia civil, com um trabalho do engenheiro Chistopher Alexander, que registrou suas experiências em resolver projetos de construções gerais, publicando os livros Notes on the Synthesis of Form, The Timeless Way of Building e A Pattern Language, descrevendo nesses livros todos os problemas que teve e como resolve-los. (HUMBERTO, 2010).

Na programação os padrões de projetos surgiram por volta dos anos 70 na linguagem Smalltalk, quando Ward Cunningham e Kent Beck desenvolveram padrões para serem aplicados na interface do usuário. Nessa mesma época, Jim Coplien desenvolveu uma lista de padrões para o C++ e Erich Gamma em sua tese de doutorado reconheceu a importância de acumular as estruturas de projetos que se repetiam com frequência no desenvolvimento do projeto. (JUNQUEIRA; COSTA; LIRA, 1998).

* + 1. **O que são Design Patterns**

Uma definição para Design Patterns pode ser: “Uma solução para um problema dentro de um contexto”. (HUMBERTO, 2010).

Dessa forma podemos definir contexto como um conjunto de situações que se repetem, onde pode ser aplicado o design patterns, a solução um conjunto de objetivos e limitações que ocorrem dentro de um contexto e a solução é uma estrutura para ser aplicada na resolução do problema. (JUNQUEIRA; COSTA; LIRA, 1998).

“Em termos de orientação a objetos, design patterns identificam classes, instâncias, seus papéis, colaborações e a distribuição de responsabilidades. Seriam, então, descrições de classes e objetos que se comunicam, que são implementados a fim de solucionar um problema comum em um contexto específico.” (JUNQUEIRA; COSTA; LIRA, 1998).

Os padrões de projetos fazem mais do que identificar a solução de um problema, eles explicam porque a solução é necessária. (JUNQUEIRA; COSTA; LIRA, 1998).

* + 1. **Por que e quando usar Design Patterns**

No processo de desenvolvimento de software OO (Orientação a objetos), os designs patterns podem ser usados de diversas formas, que segundo JUNQUEIRA, COSTA e LIRA (1998) são elas:

* Vocabulário para melhorar a comunicação entre os desenvolvedores.
* Uma documentação mais completa.
* Reduzem a complexidade do sistema através da definição de abstrações que estão acima das classes e instâncias.
* Base de experiências reutilizáveis para construção de software.
* Reduzem o tempo de aprendizado de uma biblioteca ou classe

Devemos utilizar os designs patterns em solução que se repetem com variações, por que dessa forma o reuso não se faz necessário se for somente para um determinado problema. (JUNQUEIRA; COSTA; LIRA, 1998).

Eles se aplicam muito bem quando o desenvolvedor está mais interessado na existência de uma solução do problema do que no desenvolvimento da aplicação. Isto ocorre, na maior parte das vezes, quando o domínio do problema é o foco da questão. (JUNQUEIRA; COSTA; LIRA, 1998).

Dessa forma, quando mais cedo utilizar os designs patterns em seu projeto, menor será o retrabalho em etapas mais avançadas. (JUNQUEIRA; COSTA; LIRA, 1998).

* + 1. **Tipos de Design Patterns**

Os autores do livro *Patterns of Software Architecture,* Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad e Michael Stal, definem três tipos de padrões, são eles:

* Padrão de Arquitetura
* Design pattern
* Idioma

O padrão de arquitetura representa um esquema de organização estrutural do sistema. Se preocupa com o sistema inteiro, nos seus componentes macro e propriedades globais. Afetam a estrutura e a organização do sistema por inteiro. (JUNQUEIRA; COSTA; LIRA, 1998).

O design pattern descrevem situações que se repetem, soluções para resolver problemas em contextos específicos e não afetam o sistema por inteiro.

E o idioma é um padrão de baixo nível, direcionado a uma linguagem de programação específica, exemplo a lista de padrões para o C++ visto no tópico 4.1.1. (JUNQUEIRA; COSTA; LIRA, 1998).

Conforme JUNQUEIRA, COSTA e LIRA (1998), Erich Gamma propõe um modo de categorizar os designs pattern, definindo famílias de padrões, são elas:

* Creational Patterns.
* Structural Patterns.
* Behavioral Patterns.

A Creational Patterns (Padrão Criaconais), refere-se a criação de instancias de objetos e classes. (HUMBERTO, 2010).

A Structural Patterns (Padrão Estrutural), diz respeito a organização de classes e de objetos. (HUMBERTO, 2010).

E a Behavioral Patterns (Padrão Comportamental), preocupa-se com a forma que as classes e objetos interagem e com a distribuição de responsabilidades. (HUMBERTO, 2010).

* + 1. **Design Patterns versus Frameworks**

“Um framework consiste em uma arquitetura concreta reutilizável que provê estruturas genéricas para uma família de softwares.” (JUNQUEIRA; COSTA; LIRA, 1998). Pode ser adaptado para atender várias necessidades em um determinado domínio.

Uma diferença básica de framework para bibliotecas, é que o framework determina quais objetos e métodos devem ser usados quando da ocorrência de eventos. (JUNQUEIRA; COSTA; LIRA, 1998).

Os designs patterns ao contrário dos frameworks, permitem o reuso de microarquiteturas que já possuem um vínculo, sem uma implementação concreta. (JUNQUEIRA; COSTA; LIRA, 1998).

Em geral, os frameworks são mais especializados que os designs patterns, dessa forma os designs patterns podem ser usados em um número maior de aplicações. Normalmente frameworks maduros utilizam vários designs patterns. (JUNQUEIRA; COSTA; LIRA, 1998).

* 1. **Frameworks**
  2. **Definições e conceitos do padrão MVC e as vantagens e desvantagens em sua utilização.**

O padrão MVC (Model-View-Controller) é utilizado nas aplicações para facilitar a manutenção e organização de código, ou seja ter tudo em seu lugar e cada camada com sua responsabilidade. O MVC cria uma estrutura quebrada em três camadas: Model, View e Controller. (BASTOS, 2011).

* + 1. **Model**

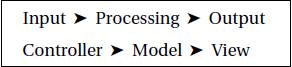
A model é responsável pela manipulação das informações e é sempre recomendado que se utilize as models para realizar consultas, cálculos e todas as regras de negócio do sistema. O modelo (model) tem acesso a toda informação, sendo ela vindo de algum bando e dados ou arquivos XML. (BASTOS, 2011).

* + 1. **View**A view é responsável por tudo o que o usuário visualiza na tela, toda interface e todas as informações são exibidas pela view. (BASTOS, 2011).
    2. **Controller**

O controller é responsável por controlar todo o fluxo de dados ou informações que passa pela aplicação. É no controller que se decide “se”, “o que”, “quando” e “onde” deve funcionar. O controller também é responsável por definir quais informações ou regras devem ser geradas ou acionadas e para aonde tem que ir as informações. É o controller que executa as regras de negócio (model) e manda as informações para a visão (view). (BASTOS, 2011).

Inicialmente o MVC foi construído com a ideia de mapear fluxo de dados. Dessa forma a entrada é o controller, o processamento é a model e a saída é a view, conforme a Figura 1.

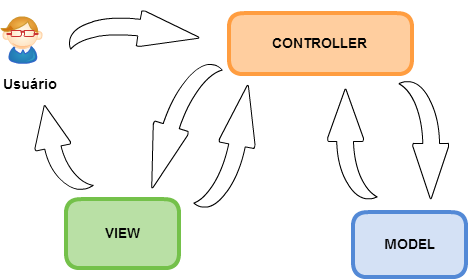
**Figura 1:** Estrutura de fluxo de dados



**Fonte**: http://www.devmedia.com.br/introducao-ao-padrao-mvc/29308

Veremos abaixo na Figura 2 como funciona o padrão MVC em uma aplicação.

**Figura 2:** Funcionamento do MVC.



**Fonte:** Elaborado pelo autor.

Por exemplo, quando o usuário se comunica com a interface (view) para realizar um cadastro, o controller fica responsável por gerenciar as entradas ou ações e mapeia essas ações em comandos que são enviados para o modelo (model) e/ou para a visualização (view) para fazer alterações adequadas. O modelo fica responsável por fazer validações, filtros e a persistência dos dados. Por fim a view, que fica responsável para apresentar as informações ao usuário. Tudo que ela faz é receber instruções ou dados do controller e exibir. Voltando para a view, o ciclo começa novamente. (MEDEIROS).

* + 1. **Vantagens e desvantagens na utilização do padrão MVC.**

Existem algumas vantagens e desvantagens na utilização do padrão MVC para criação de aplicações, porém são muito mais vantagens do que desvantagens.

Conforme BASTOS (2011), as vantagens em se utilizar o MVC são:

* Separação muito clara entre as camadas de visualização e de regras de negócio.
* Manutenção do sistema ou aplicação se torna muito mais fácil.
* Reaproveitamento de código.
* As alterações na camada de visão não afetam as regras de negócio já implementadas na model.
* Permite o desenvolvimento, teste e manutenção de forma separada entre as camadas.
* Melhor organização do projeto.
* Melhora o entendimento do projeto, necessário para novos desenvolvedores que não participaram na criação do mesmo.
* Torna a aplicação escalável.

As desvantagens em se utilizar o MVC são:

* Em sistemas de baixa complexidade, o padrão MVC pode criar uma complexidade maior, ou seja, não é aconselhado utilizar em projetos pequenos.
* Exige muita disciplina dos desenvolvedores em relação a separação em camadas.
* Requer um tempo maior para modelar o sistema.

Como podemos ver, existem muito mais vantagens em se utilizar o padrão MVC para o desenvolvimento de aplicações do que desvantagens. Ideal para quem quer um código bem organizado e de fácil entendimento. (BASTOS, 2011).

Para melhor organizar o código JavaScript, foram criados vários frameworks com a ideia do padrão MVC.

No próximo tópico veremos o Angular.js, um framework JavaScript MVC.

* 1. **História do JavaScript.**

O JavaScript é uma linguagem de programação foi desenvolvida originalmente por Brendan Eich da Netscape com o nome de Mocha, posteriormente foi mudado para LiveScript e atualmente JavaScript. LiveScript foi lançada pela primeira vez na versão beta do navegador NetScape 2.0 em setembro de 1995. (STENZEL, 2013).

A mudança de nome para JavaScript se deu na época em que a NetScape adicionou suporte à tecnologia Java em seu navegador. Com essa escolha final, causou confusão dando impressão de que a linguagem foi baseada em Java, sendo que essa escolha foi caracterizada por muitos como uma estratégia de marketing da NetScape para aproveitar a fama do recém-lançado Java. (STENZEL, 2013).

“JavaScript foi a primeira linguagem a ser tratada como uma extensão da HTML e que trouxe capacidade de processamento cliente-side”. (CARVALHO, 2001, p. 2).

No começo muitos desacreditaram da linguagem, pois a mesma tinha como principal alvo o público leigo, mas com o passar dos tempos o JavaScript tem se transformado na linguagem de programação mais popular da web. (STENZEL, 2013).

Com a chegada do Ajax, o JavaScript teve sua popularidade aumentada, recebendo mais atenção dos profissionais. Com esse grande aumento, houve grande proliferação de bibliotecas e frameworks e o aumento no uso do JavaScript fora do ambiente de navegadores bem como o uso de plataformas de JavaScript server-side (lado servidor). (STENZEL, 2013).

As principais características do JavaScript, é ter tipagem dinâmica, funções internas, dentre outras que será abordado no próximo tópico.

Segundo CARVALHO (2001, p. 2), algumas possibilidades que o JavaScript traz, que outras linguagens como o HTML e CSS trazem, não ultrapassam as do JavaScript:

* Efetuar contas matemáticas.
* Gerar páginas com aparências que podem ser definidas em tempo de execução.
* Tratar determinados eventos, podendo interagir com o usuário através desse tratamento.
* Manipular janelas dos navegadores.
* Modificar propriedades da página dinamicamente.

JavaScript é a linguagem mais utilizada pelos desenvolvedores da Web e sua entrada no mercado foi levada em consideração pela ECMA (European Computer Manufactures Association), uma entidade normativa europeia. (CARVALHO, 2001, p. 2).

* + 1. **Características do JavaScript.**

O JavaScript possui programação imperativa, que é por sequência de ordens, chamada de comandos ou instruções (ALBUQUERQUE, 2010) e a estruturada, que é um método para combinar as estruturas de controle formadas por blocos de maneira organizada e que traduza o problema a ser resolvido. (SATO, 2009).

Possuiu tipagem dinâmica, ou seja, não tem declaração de tipo de dados. É baseada em objetos, que no JavaScript são *arrays* associativos. JavaScript inclui uma função chamada *eval*, que consegue executar em tempo de execução comandos da linguagem. Java script tem funções ‘internas’ ou ‘aninhadas’ que são funções definidas dentro de outras funções. JavaScript permite que funções aninhadas sejam criadas com o escopo estático, no momento de sua definição e possui o operador “()” para invocá-las em outro momento, ou seja, se no código-fonte uma estrutura está aninhada em outra, a de dentro pode acessar variáveis na de fora. (PRADO, 2014).

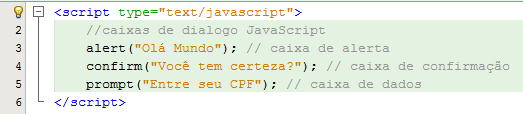
Talvez o principal aspecto seja que o JavaScript é *case-sensitive,* ou seja, existe diferença entre letras maiúsculas e minúsculas. Dessa forma, “document” é diferente de “Document”, que para o HTML isso não ocorre. (CARVALHO, 2001, p. 16).

Outra característica que foi herdado do JAVA, é a utilização de ponto e vírgula no final de cada comando, por exemplo: *var numero = 1234;.* Apesar de ter herdado, não é uma regra da linguagem, mas sua utilização pode evitar problemas e erros na execução de scripts. (CARVALHO, 2001, p. 16).

Segundo SILVA (2010, p. 43), comentários também estão à disposição dos desenvolvedores. Existe três tipos de comentários: comentários de uma linha (variante 1), comentário em linha única (variante 2) e os comentários de múltiplas linhas.

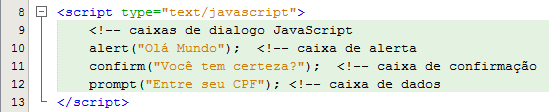
Vejamos abaixo nas Figuras 3, 4 e 5, exemplos de utilização de comentários.

**Figura 3**: Exemplos de comentários linha única (variante 1).



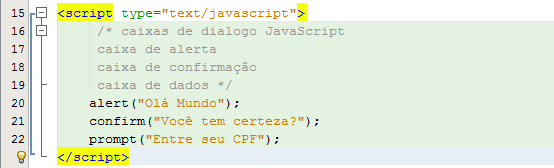
**Fonte:** SILVA (2010, p. 43).

**Figura 4**: Exemplos de comentários linha única (variante 2).



**Fonte:** SILVA (2010, p. 43).

**Figura 5**: Exemplos de comentários múltiplas linhas.



**Fonte:** SILVA (2010, p. 43).

Use duas barras “//” para iniciar um comentário em linha única ou coloque “/\* \*/” para múltiplas linhas. (SILVA, 2010, p. 43).

A variante 2, para comentários de uma linha, usa uma sintaxe semelhante a do HTML, com a diferença de que não é necessário seu fechamento, como na Figura 6 a seguir:

**Figura 6**: Exemplo de comentário para marcação HTML e de JavaScript;



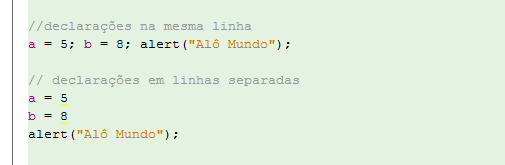
**Fonte**: SILVA (2010, p. 43).

Utilize a variante 1 para os comentários de uma linha. (SILVA, 2010, p. 43).

* + 1. **Declarações**

Cada uma das instruções de um script constitui uma declaração independente. Para separar uma declaração de outra existem duas formas. Conforme as Figuras 7, podemos separar por ponto e vírgula e em linhas diferentes. (SILVA, 2010, p. 44).

**Figura 7**: Formas de declaração.



**Fonte:** SILVA (2010, p. 44).

Conforme abordado anteriormente, o ponto e vírgula não é obrigatório, mas é recomendado a utilização para não haver erros. Para o caso de declaração na mesma linha, a utilização de ponto e vírgula se faz necessário.

* + 1. **Literais**

Segundo SILVA (2010, p. 45), a palavra literal para o JavaScript quer dizer qualquer dado que não seja variável, ou seja, que tenha seu valor fixo.

No exemplo da Figura 8, os valores 50 e Alô Mundo, são literais.

**Figura 8**: Exemplo de dados literais



**Fonte**: SILVA (2010, p. 45)

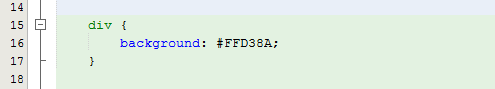
Existem seis tipos de dados literais no JavaScript, são eles:

* Inteiros
* Decimais
* Booleanos
* Strings
* Arrays
* Objetos

Para a sintaxe o JavaScript, os inteiros são números em base decimal (base 10), hexadecimais (base 16) ou octal (base 8).

A base 10 é a mais conhecida, usadas na matemática no conjunto Z que os alunos aprendem no ensino médio, por exemplo: Z = {..., -2, -1, 0, 1, 2, ...}, ou seja, conjunto dos números inteiros positivos e negativos incluindo o número zero. (SILVA, 2010 p. 45). A base hexadecimal é muito utilizada no CSS, nas declarações de cores, conforme a Figura 9. Seus símbolos são de 0 a 9 juntamente com as 6 primeiras letras do alfabeto, de A a F.

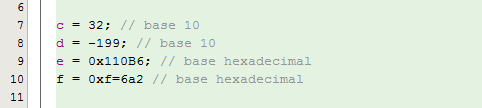
**Figura 9**: Exemplo de hexadecimal no CSS



**Fonte**: SILVA (2010, p. 46).

Para utilização de hexadecimal nos arquivos JavaScript é necessário utilizar antes o 0x (zero xis), conforme a Figura 10.

**Figura 10:** Exemplo de declarações usando literal inteiro.

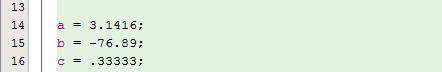


**Fonte:** SILVA (2010, p. 46).

Note que para a sintaxe na base hexadecimal, não é necessário a utilização de letras somente maiúsculas ou minúsculas. (SILVA, 2010, p. 46).

Os literais decimais, são os números constituídos por um inteiro e um número fracionário, separados por um ponto (.), por exemplo a Figura 11.

**Figura 11**: Literais decimais no JavaScript

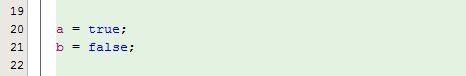


**Fonte:** SILVA (2010, p. 46).

O JavaScript suporta notação cientifica, para escrever tanto literais inteiro como fracionários. As notações cientificas são maneiras de representar números compostos por muitos algarismos e consiste em acrescentar a letra E ou e a um número para indicar expoentes de 10. (SILVA, 2010, p. 46).

Os booleanos, são representados no JavaScript pelas palavras-chave *true* e *false* (minúsculas)*,* definindo condições verdadeiras ou falsas, respectivamente. (SILVA, 2010, p. 497). Na figura 12 vemos como o JavaScript trata os booleanos.

**Figura 12**: Booleanos no JavaScript

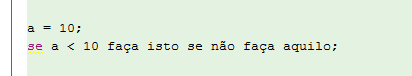


**Fonte:** SILVA (2010, p. 47).

São muito utilizados para controle, testar a validade de uma determinada condição. Dessa forma permite ao script tomar uma decisão baseando na condição de verdadeiro ou false retornada pela condição feita pelo desenvolvedor. (SILVA, 2010, p. 43).

Vejamos abaixo na Figura 13, o uso de booleano.

**Figura 13**: Utilização de booleano;



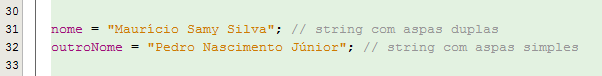
**Fonte**: SILVA (2010, p. 47).

Transformando essa lógica em sintaxe JavaScript, temos uma condição que retornara *true* caso seja menor que 10, e *false* caso contrário.

Strings são sequência de caracteres, já para o JavaScript, são um conjunto de zero ou mais caracteres entre aspas duplas (“”) ou simples (‘’). (SILVA, 2010, p. 47).

Na Figura 14 abaixo, podemos observar como o JavaScript trata as Strings.

**Figura 14**: Strings no JavaScript.



**Fonte**: SILVA (2010, p. 47).

Podemos também utilizar caracteres especiais para formatar uma string, por exemplo a Figura 15.

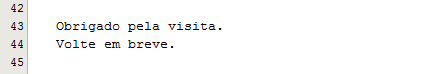
**Figura 15**: Formatação de Strings utilizando caracteres especiais no JavaScript.



**Fonte**: SILVA (2010, p. 47).

Ao inserir o caractere especial \n na string, ele pula (quebra) uma linha onde estivar localizado, dando o resultado conforme a Figura 16.

**Figura 16**: Resultado da utilização do \n



**Fonte**: SILVA (2010, p. 47).

Na Tabela 1 abaixo, observe alguns caracteres especiais da linguagem e a sintaxe geral para caracteres Latin-1 e Unicode (bibliotecas de caracteres).

**Figura 15**: Tabela de caracteres especiais JavaScript.

|  |  |
| --- | --- |
| **Caractere** | Descrição – Caractere Unicode hexadecimal |
| \b | Backspace - \u0008 |
| \f | Form feed - \u000C |
| \n | Nova linha - \u000A |
| \r | Retorno do carro - \-u000D |
| \v | Tabulação vertical - \u000B |
| \t | Tabulação horizontal - \u0009 |
| \’ | Apostrofo 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 dois dígitos hexadecimais de 00 a FF |
| \uXXXX | Caractere Unicode expresso por quatro dígitos hexadecimais |

**Fonte**: SILVA (2010, p. 48).

Nas três últimas linhas consta a sintaxe geral para representação de caracteres em string em hexadecimal e octal. O sistema octal está em desuso, então deve ser evitado. (SILVA, 2010, p. 47).

Os arrays, são conjunto de zero ou mais valores, que são separados por vírgula e estão dentro de colchetes ([]). Todo array, sem definir sua posição começa com índice zero, depois um, assim sucessivamente. (SILVA, 2010, p. 51). Vejamos abaixo na Figura 17, como é declarado um array no JavaScript.

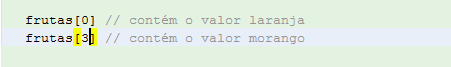
**Figura 17:** Declaração de array JavaScript.



**Fonte**: SILVA (2010, p. 51).

Na Figura 18 abaixo, mostra um exemplo de como pegar um valor do array, que consiste em pegar o nome do array, seguido de um índice entre colchetes. (SILVA, 2010, p. 51).

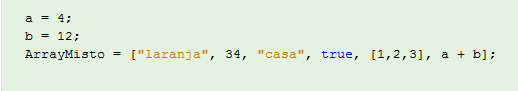
**Figura 18:** Pegar o valor em uma posição do array



**Fonte:** SILVA (2010, p. 51).

Além disso, um array pode conter vários tipos de dados, expressões, e outros arrays, como mostra a Figura 19 abaixo.

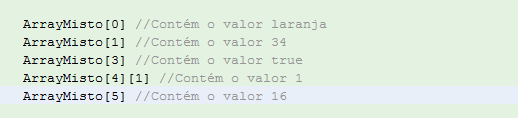
**Figura 19:** Sintaxe de um array misto



**Fonte:** SILVA (2010, p. 51).

Agora basta fazer a extração dos dados do array, conforme a Figura 20.

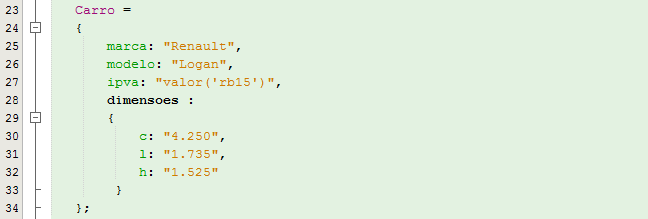
**Figura 20:** Pegar o valor em um array misto



**Fonte:** SILVA (2010, p. 51).

E finalmente os objetos, que para o JavaScript, são conjuntos de pares nome/valor separador por vírgula e envolvidos por chaves ({}). A relação de nome/valor pode ser da forma, objeto e seu valor, um método do objeto e seu valor e mesmo outro objeto. Para criar um objeto, temos a seguinte sintaxe, conforme Figura 21. (SILVA, 2010, p. 52).

**Figura 21:** Sintaxe para criação de um objeto



**Fonte:** SILVA (2010, p. 52).

O objeto que foi criado chamado de Carro, possui três propriedades, a *marca*, o *modelo* e a *dimensão*. Dentro da propriedade dimensão temos mais objeto com três propriedades, que são *c*, *l* e *h*, que definem o comprimento, a largura e a altura do carro, e também temos um método com nome de IPVA, que é uma função chamada *valor(),* que aceita um argumento de cálculo *rb15*, com intenção de calcular o IPVA do carro. (SILVA, 2010, p. 52).

Para percorrermos um objeto, utilizamos a seguinte sintaxe: *propriedade.valor*, conforme Figura 22 a seguir.

**Figura 22:** Sintaxe para percorrer um objeto



**Fonte:** SILVA (2010, p. 52).

As palavras reservadas do JavaScript segundo SILVA (2010, p. 56), as palavras-chaves são conforme mostra a Tabela 3.

**Tabela 3:** Palavras-chave do JavaScript.

|  |  |  |  |
| --- | --- | --- | --- |
| break | else | new | var |
| case | finally | return | void |
| catch | for | switch | while |
| continue | function | this | with |
| default | if | throw | delete |
| in | try | do | instanceof |
| typeof |  |  |  |

**Fonte:** SILVA (2010, p. 56).

Antes de começarmos a ver sobre o Angular.js, um framework JavaScript MVC, precisamos conhecer um pouco do jQuery, uma biblioteca para auxiliar no desenvolvimento de aplicações JavaScript, (SILVA, 2010, p. 25), para podermos fazer uma pequena comparação com o Angular.js.

* 1. **Definições e conceitos de jQuery**

Segundo SILVA (2013, p. 25), jQuery é uma biblioteca JavaScript. Essa biblioteca foi criada por John Resig, é um software livre e aberto, ou seja, sua utilização é gratuita tanto para projetos pessoais quanto para comerciais. Por ser um software livre, existe uma enorme comunidade que apoia e contribuem com novas ideias, scripts, plug-ins, extensões, não só para acrescentar na biblioteca, mas para incrementar técnicas de desenvolvimento jQuery. (SILVA, 2013, p. 31).

John escreveu em seu livro *jQuery in Action*, uma frase que diz o principal foco da biblioteca jQuery. “O foco principal da biblioteca jQuery é a simplicidade. Por que submeter os desenvolvedores ao martírio de escrever longos e complexos códigos para criar simples efeitos?”. (SILVA, 2013, p. 25).

* + 1. **Histórico**

No dia 22 de agosto de 2005, John muito frustrado com a maneira de escrever códigos em JavaScript, escreveu em seu blog um artigo relatando essas frustrações. Nesse artigo escreveu exemplos no qual propôs o uso de seletores CSS, por exemplo, para pegar um *id* de um elemento é usado *$(‘#id\_elemento’),* para classes é usado *$(‘.classe’)*, para simplificar e dar versatilidade ao código. Nessa ocasião foi lançada a ideia, que mais pra frente resultaria na biblioteca jQuery. (SILVA, 2013, p. 26).

No ano de 2006 foi o marco da criação do primeiro plug-in para a biblioteca. Nesse ano ainda foram lançadas as versões 1.0, 1.0.1, 1.0.2, 1.0.3 e 1.0.4, da versão XML da biblioteca. Hoje segundo o site oficial do jQuery (https://jquery.com/), está na versão 2.1.3, mas não suportado para Internet Explorer (IE) nas versões 6, 7 e 8.

* + 1. **Utilização e conformidade com Padrões Web.**

O jQuery é utilizado para dar interatividade e dinamismo para as páginas web, proporcionando ao desenvolvedor funcionalidades necessárias para criação de scripts que tendam a incrementar usabilidade, acessibilidade e o design, enriquecendo a experiência do usuário. (SILVA, 2013, p. 28).

Segundo SILVA (2013, p. 28), utilize jQuery para os seguintes casos:

* Adicionar efeitos visuais e animações
* Acessar e manipular DOM
* Busca dados no servidor sem recarregar a página.
* Interatividade
* Alterar conteúdos
* Modificar apresentações e estilização
* Simplificar tarefas do JavaScript

O jQuery foi criado para ser uma biblioteca com conformidade dos Padrões Web, sendo compatível com qualquer navegador ou sistema operacional e dando suporte total para CSS3, sem se preocupar se o navegador suporta o seletor para CSS. Mas isso não significa que todo código que for escrito em jQuery, resulta em um documento válido dos Padrões Web. Cabe ao desenvolvedor escrever seus códigos em conformidade com os padrões. (SILVA, 2013, p. 29).

* + 1. **Caracteristicas**

Segundo o SILVA (2013, p. 30), o jQuery possui as seguintes características:

* Utiliza seletores CSS para localizar elementos da estrutura de marcação HTML.
* Possui arquitetura compatível com instalação de plug-ins e extensões em geral.
* Não leva em consideração a incompatibilidade de Renderização entre os navegadores.
* Não há necessidade de construção de loops para localização de elementos no documento.
* Admite criação e inserção de novas funcionalidades na biblioteca.

Essas caracteristicas possibilitam o jQuery ser uma biblioteca poderosa e compacta para tornar o desenvolvimento compacto e simples. (SILVA, 2013, p. 30).

* + 1. **Instalação**

A biblioteca jQuery é um simples arquivo ,js, por exemplo, *minha\_biblioteca.js*, que deverá ser chamado na página onde você quer aplicar efeitos. Não é necessário instalar nenhum arquivo. (SILVA, 2013, p. 31).

Segundo SILVA (2013, p. 31). Existem duas formas de chamar a biblioteca em uma página web:

* A partir de um arquivo público remoto conhecido como CDN (Domain Name System).
* A partir de um arquivo remoto público.
  + - 1. **A partir de um arquivo público remoto (arquivo local)**

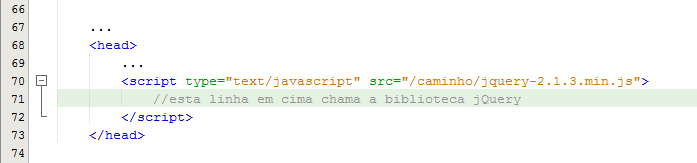
“O Google e a Microsoft e a biblioteca jQuery, onde são hospedadas as diferentes versões da biblioteca para acesso público”. (SILVA, 2013, p. 31).

Basta fazer o download gratuito do arquivo e depois só linká-las (chama-las) nas páginas que será utilizado a biblioteca.

Segundo o Site oficial JQUERY a biblioteca se encontra na versão 2.1.3.

Na Figura 23 abaixo, mostra como deve ser linkado a biblioteca para se utilizar nas páginas web.

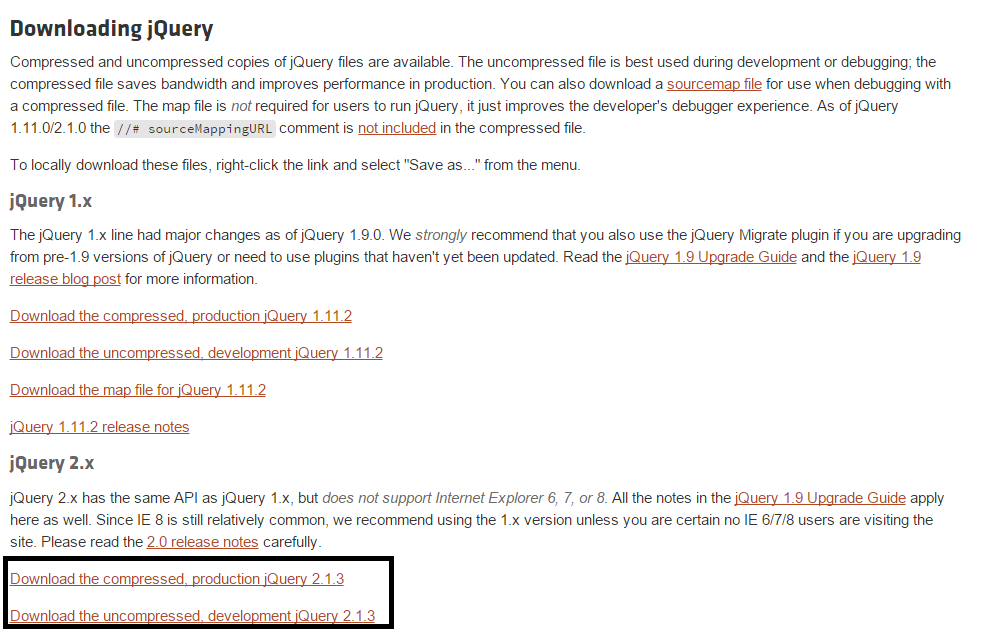
**Figura 23:** Forma de linkar a biblioteca jQuery arquivo local



**Fonte:** SILVA (2013, p. 32).

Na Figura 24 abaixo, observe que existem dois tipos de apresentações da biblioteca para o download.

**Figura 24:** Formas de download da biblioteca jQuery



**Fonte:** SILVA (2013, p. 32).

O link para download chamado “*Download the uncompressed, development jQuery 2.1.3*”, é um arquivo para use em testes, estudos e desenvolvimento da própria biblioteca. É um arquivo onde seu texto do script foi totalmente comentado e com espaço entre as linhas de código, para facilitar o entendimento, a leitura e análise do desenvolvedor. (SILVA, 2013, p. 32).

O link chamado “*Download the compressed, prodution jQuery 2.1.3*”, não tem em seu código, comentários, espações entre as linhas e sem declarações, dessa forma deixando o código mais difícil de entendimento, pois o arquivo é sem quebra de linha, ou seja, código escrito tudo em uma única linha. (SILVA, 2013, p. 32).

* + - 1. **A partir de um arquivo remoto público (arquivo remoto)**

O Google, disponibiliza uma área chamada “*Google Hosted Libraries - Developer's Guide*”, com as bibliotecas JavaScript e suas formas para fazer a *linkagem*, para utilizar em suas páginas web. (https://developers.google.com/speed/libraries/devguide).

Há duas formas de linkar a biblioteca. Na Figura 25 mostra a primeira opção, que faz o uso do elemento *script,* da mesma forma que foi mostrado para chamar a biblioteca no arquivo local, conforme a Figura 24. (SILVA, 2013, p. 33).

**Figura 25:** Forma de linkar a biblioteca jQuery com arquivo remoto



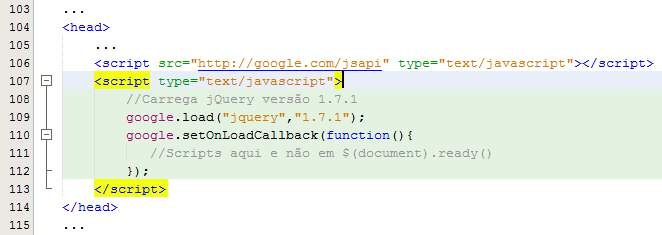
**Fonte:** SILVA (2013, p. 33).

No exemplo mostrado, está sendo feito uma *linkagem* para a versão 2.1.3 “*minified*” (compactada). (SILVA, 2013, p. 33).

A segunda forma, utiliza o método *google.load()* para carregar a biblioteca. (SILVA, 2013, p. 33).

Esse método foi desenvolvido pelo Google, chamando esse método é possível carregar as versões da biblioteca de uma maneira mais flexível do que o exemplo da Figura 25. Na Figura 26, observamos como é feita a *linkagem* com esse método. (SILVA, 2013, p. 34).

**Figura 25:** Forma de linkar a biblioteca jQuery com google.load()



**Fonte:** SILVA (2013, p. 34).

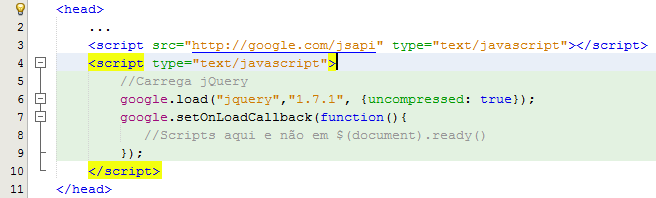
Na linha 106, destina-se a definir métodos e funções necessárias para carregar as bibliotecas da API do Google. (SILVA, 2013, p. 34).

A linha 109, fica responsável por carregar uma biblioteca da API, que nesse exemplo irá carregar a biblioteca jQuery, versão 1.7.1. (SILVA, 2013, p. 34).

Já na linha 110, quando se utiliza o *google.load()* é recomendado usar aquela sintaxe. (SILVA, 2013, p. 34).

Podemos observar que existem dois parâmetros passados na função do *google.load()*. O primeiro parâmetro é passado o nome da biblioteca que se deseja utilizar e o segundo é a versão que ver quer utilizar a biblioteca, que por padrão ela está na sua versão compactada. Além desses dois parâmetros vistos, podemos ainda utilizar um terceiro parâmetro, neste caso sendo opcional. Essa terceira opção fica responsável por carregar uma versão não compactada caso acha interesse do desenvolvedor. Na Figura 26, veremos um exemplo da utilização do terceiro parâmetro. (SILVA, 2013, p. 34).

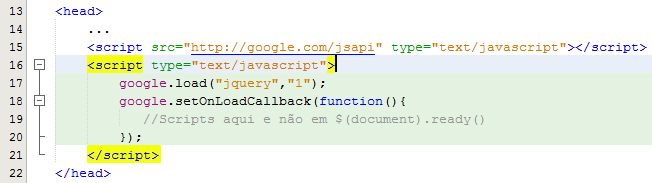
**Figura 26:** Como utilizar o terceiro parâmetro do método *google.load()*



**Fonte:** SILVA (2013, p. 35).

Podemos também fazer que o *google.load()* se torne dinâmico. Vejamos o exemplo na Figura 27.

**Figura 27:** Utilizar o *google.load()* dinamicamente



**Fonte:** SILVA (2013, p. 36).

No exemplo da Figura 27, cada lançamento da versão começando com o número 1, a biblioteca será atualizada em seu script, mas como a biblioteca se encontra na versão 2.1.3, basta você colocar o número 2, e sua página estará com a versão mais atual. (SILVA, 2013, p. 36).

A Microsoft também disponibiliza a biblioteca jQuery a partir da versão 1.3.2 para uso público, em suas versões “*uncompressed*” e “*mini*”. Conforme consta na documentação no site da Microsoft no link *http://www.asp.net/ajax/cdn*, para chamar a biblioteca para uma página web, utiliza-se o código conforme a Figura 28.

**Figura 28:** Linkar biblioteca jQuery com servidor Microsoft



**Fonte:** SILVA (2013, p. 36).

Agora que estudamos um pouco sobre jQuery, podemos ver sobre o Angular.js, e fazer algumas comparações entre os dois.

* 1. **Definições e conceitos do Angular.js framework JavaScript MVC**

O Angular.js é um framework *Client-side* (rodapé) e que foi criado com JavaScript, fundamental para aplicações web de uma página SPA (single page applications) e até mesmo aplicações.

* 1. **Conceitos de DOM e como os navegadores interpretam**

DOM (Document Object Model), criado pela W3C1 (World Wide Web Consortium), representa a hierarquia da página, que nos permite mexer na estrutura do documento. Permite ver qual atributo está associado a cada objeto e como manipulá-los. (FRANKLIN, 2011).

DOM subdivide-se em três subconjuntos, o DOM Core, DOM HTML e DOM XML. (SILVA, 2010, p. 335).

DOM Core, é uma API (Application Programming Interface), que define um conjunto de objetos e interfaces para manipular e acessar os objetos de um documento. (SILVA, 2010, p. 383).

“DOM HTML é uma representação da estrutura do documento HTML”. (SILVA, 2010, p. 326).

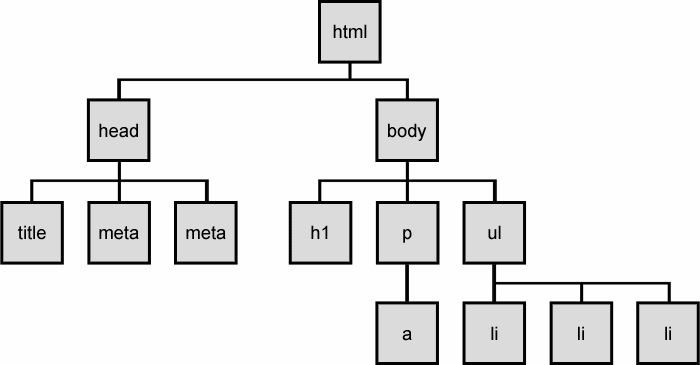
E DOM XML, segundo a W3SCHOOLS, é um padrão para acessar e manipular documentos XML.

“O DOM XML define os objetos e propriedades de todos os elementos XML, bem como a métodos (interface) para acessá-los.” (W3SCHOOLS)

“Em outras palavras: O DOM XML é um padrão para como chegar, alterar, adicionar ou excluir elementos XML.” (W3SCHOOLS)

Na Figura 3 abaixo vamos ver como o DOM em sua forma e como é reconhecido pelos navegadores.

**Figura 3:** DOM em sua forma.



**Fonte:** http://tableless.com.br/tenha-o-dom/

Na Figura 3, mostra a estrutura de uma arvore DOM, que segundo FRANKLIN (2011) não é a estrutura que vemos no navegador, é somente uma base para uma outra estrutura, que é o realmente um navegador (browser) monta na tela, a Arvore de Renderização – Render Tree ou podemos dizer o produto final do HTML.

A base *class*, chamado de Node.h é a base para todo os nós da árvore DOM. O Node.h possuiu várias categorias e as necessárias para obter o código no navegador são os nós de documentos, elementos e texto. (FRANKLIN, 2011).

O documento é o nó mais importante e é divido em três classes diferentes. O *document* usado por todos os documentos XML e outros que não sejam SVG (Scalable Vector Graphics), que é um formato aberto de imagem vetorial 2D, onde temos pontos ligados entre si. (HECKERT, 2008). O *HTMLDocument* que cuida documentos HTML e o ***SVGDocument* responsável pelos documentos SVG e outros documentos herdados da classe Document.**

**O nó elemento, são todas as *tags* que estão nos arquivos HTML e XML, que se transformam em nós da árvore DOM.**

**E o nó texto, é o texto que vai entre os elementos.** (FRANKLIN, 2011).

A única diferença entre o DOM e a Render Tree, é que a Render Tree possui objetos que não possuem nós no DOM, por exemplo folhas de estilos, script.

Para criação da Render Tree, existem três passos básicos, são eles:

* Attachment: Após a criação do DOM o navegador chama o método attach, que adiciona primeiro os estilos na árvore DOM e começa a estilizar a página.
* RenderStyle.h: durante o processo do attach, é criado o método RenderStyle.h, onde vai guardar objetos de referência com cada uma das propriedades CSS do documento. Quando os nós são criados no DOM, eles são verificados no documento de CSS, caso existam propriedades, elas são aplicadas.
* CSS Box Model: é usado para posicionar os elementos na página. Como a utilização de margens, bordas, entre outros. (FRANKLIN, 2011).

Todos os navegadores possuem uma lista de elementos que suportam. Dessa forma quando sua aplicação possui as tags presentes nessa lista, a árvore DOM é montada e começa o processo de criação da Render Tree. Um grande problema em relação a isso, é que todos os navegadores possuem uma lista diferente de elementos. O IE (Internet Explorer) é o navegador que mais apresenta problemas. (FRANKLIN, 2011).

Conforme FRANKLIN (2011), os elementos que estão fora dessa lista são conhecidos ou tratados como elementos desconhecidos. Para esses elementos a ideia é não estilizar. Mas caso queira estilizar o elemento você deve colocá-lo no nó acima, para fazer com que ele herde o estilo.

* 1. **Linguagem PHP**
  2. **Definição de API´s**
  3. **Definição de REST.**
  4. **Ferramentas**
     1. **Netbeans**

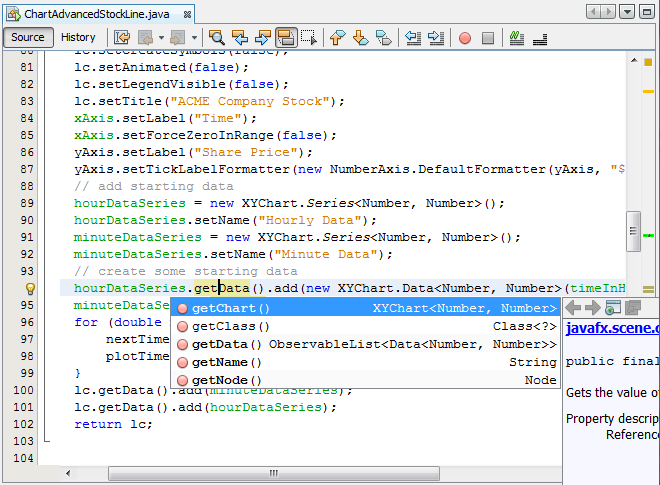
Segundo o NETBEANS, o Netbeans IDE (Integrated Development Environment) ou “Ambiente Integrado de Desenvolvimento”, é o IDE oficial do Java8. IDE é um programa para computador onde reúne ferramentas e caracteristicas para apoio ao desenvolvimento de software.

“Netbeans IDE pode ser instalado em todos os sistemas operacionais que suportam Java, por exemplo, Windows, Linux, Mac OS X. O próprio Netbeans é escrito em Java”. (NETBEANS).

A comunidade do Netbeans é grande e ativa, muitos usuários estão desenvolvendo novos plug-ins o tempo todo porque a plataforma é extensível e tem API’s bem documentadas. (NETBEANS).

Um IDE, não é apenas um editor de texto, ele permite ao desenvolvedor, facilita refatorar o código com uma serie de ferramentas poderosas, fornece modelos de códigos, dicas de codificação e geradores de código. A Figura 29, mostra a inteligência de edição de código. Você pode ir digitando e ele mostra as dicas de codificação ou você digita alguma coisa e tecla “*ctrl + space*”, que ele irá mostrar dicas.

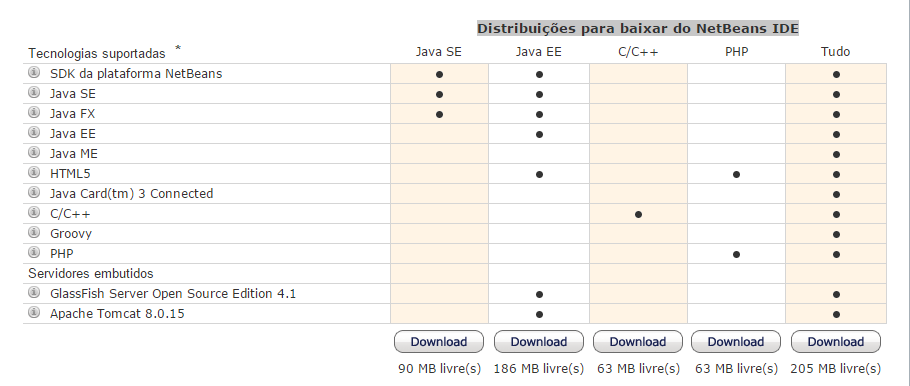
**Figura 29:** Rápido & inteligente Edição de código



**Fonte:** https://netbeans.org/downloads/

O Netbeans IDE tem suporte a várias linguagens, por exemplo, conforme a Figura 30. (NETBEANS).

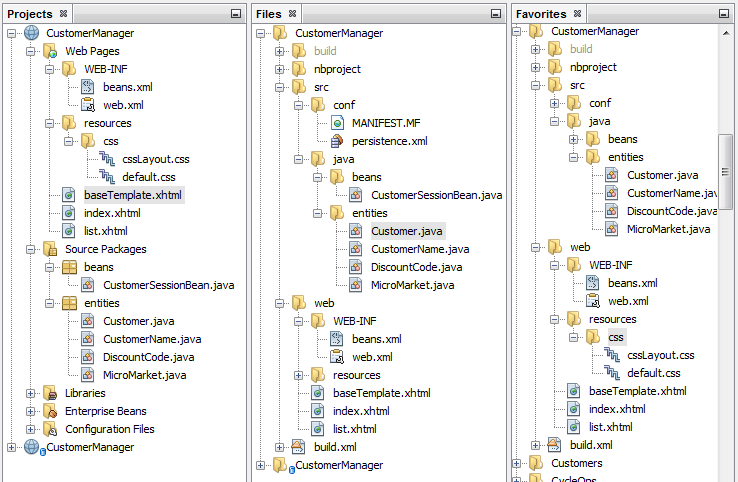
**Figura 30:** Distribuições para baixar do Netbeans IDE



**Fonte:** https://netbeans.org/downloads/

Para mantar um código com uma visão clara com milhares de pastas e arquivos e com milhões de linhas de código não é uma tarefa fácil. Para facilitar esse problema, o Netbeans oferece diversos pontos de vista dos seus dados, permitindo você ir a qualquer pasta de uma maneira fácil e rápida. A Figura 31, mostra como é feita a hierarquia dos arquivos dentro do Netbeans.

**Figura 31:** Hierarquia dos dados do Netbeans



**Fonte:** https://netbeans.org/downloads/

Além disso o Netbeans permite que você coloque interrupções nos seus códigos fontes, adicione inspeções de campo.

“O Netbeans Profiler presta assistência especializada para otimizar a velocidade e memória do seu aplicativo uso, e torna mais fácil para construir aplicações de aplicativos Java SE, JavaFX e Java EE. Netbeans IDE inclui um depurador visual para aplicativos Java SE, permitindo que você depurar interfaces de usuário, sem olhar no código-fonte”. (NETBEANS).

Netbeans é *open source* (código aberto), ou seja, são livres para uso comercial e não comercial, o código fonte está disponível para qualquer pessoa reutilizar sob os termos de licença, indivíduos ou empresas podem contribuir com o código para o projeto.

“Com exceção dos conteúdos que especifica explicitamente a licença sob a qual é disponibilizado, o software e outro conteúdo deste site está licenciado sob a Common Development and Distribution License, o que está em conformidade com o Open Source Definition”. (NETBEANS).

O download mais recente que está disponível aos usuários é o Netbeans IDE 8.0.2, que é uma atualização do Netbeans IDE 8.0. Fornece analisadores de código, editores para trabalhar com as mais recentes tecnologias Java 8 – Java SE 8, Java ME, novas ferramentas para HTML5, em particular para o Angular.js, melhorias para PHP e suporte C/C++.

* + - 1. **Instalação**

Para começarmos a instalação devemos ver os Softwares necessários. Segundo o site do Netbeans, são eles:

* O Java SE Development Kit (JDK) 7 ou JDK 8 para instalar o Netbeans IDE. Para baixar a versão mais recente do JDK 7 e JDK 8, acesse <http://www.oracle.com/technetwork/java/javase/downloads>.
* Para o PHP e C/C++ é necessário o Java Runtime Environment (JRE) 7 ou 8.
* JDK 8 é necessário para usar JavaFX 8 recursos no NetBeans IDE 8.0.2.

Para instalar o JDK 8 no Windows, basta fazer o download, o arquivo vira com a extensão *.exe*, e basta dar dos clicks para executá-lo. Já para o arquivo Linux, para utilizar o PHP, basta executar o comando *sudo apt-get install openjdk-7-jdk* no terminal para instar o JDK 7.

Agora vamos à instalação para Windows e Linux, segundo as instruções do NETBEANS. O NETBEANS fornece os passos para a instalação para Mac Os X também.

1. Se o download for para Windows, o arquivo terá a extensão *.exe,* basta dar dos cliques para executá-lo. Se for para Linux, o arquivo terá a extensão *.sh*, bata entrar na pasta que foi salvo o download, por exemplo *cd Download,* e executar o comando *sudo sh nome\_arquivo.sh* e teclar *enter.*
2. Se você baixou o pacote **Tudo** ou **Java EE bundle**, como foi visto na Figura 30 anteriormente, você pode personalizar a sua instalação. Execute as seguintes etapas na página de boas-vindas do assistente de instalação:

* Clique em Personalizar.
* Na caixa de diálogo Personalizar a instalação, faça suas seleções.
* Clique em OK.

1. Na página de boas-vindas do assistente de instalação, clique em Avançar.
2. Na página Contrato de Licença, revise o contrato de licença, clique na caixa de seleção para aceitar e clique em Avançar.
3. Na página Contrato de Licença JUnit, decida se você deseja instalar o JUnit e clique na opção apropriada, clique em Avançar.
4. Aceite o diretório padrão de instalação do Netbeans IDE ou especifique outro diretório.
5. Aceite a instalação padrão do JDK para usar com o Netbeans IDE ou selecione uma instalação diferente na lista drop-down.
6. Se a página de instalação do GlassFish Server Open Source 4.1 abre, aceite o diretório padrão de instalação ou especifique outro local de instalação.
7. Na página Resumo, faça o seguinte:

* Verifique se a lista de componentes a serem instalados está correta.
* Selecione a caixa de seleção Verificar Atualizações para se você quiser verificar o Centro de Atualização para possíveis atualizações e tem a biblioteca JUnit instalado durante a instalação (desde que você aceite a licença na etapa 5.)
* Verifique se você tem espaço suficiente no sistema para a instalação.

1. Clique em Instalar para iniciar a instalação.
2. Na página Instalação concluída, fornecer dados de uso anônimas se desejar, e clique em Concluir.

Após executar esses passos, seu Netbeans está pronto para ser usado.

* + 1. **PostgresSql e pgAdmin III**

1. **Procedimentos metodológicos**

Instalação da IDE Netbeans para criação da aplicação.

1. **Cronograma**

Cronograma para realização do Projeto, conforme Tabela 4.

**Tabela 4:** Cronograma das atividades.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Atividades** | **Mar** | **Abr** | **Mai** | **Jun** | **Jul** | **Ago** | **Set** | **Out** | **Nov** |
| Pesquisa e desenvolvimento do projeto | **X** |  |  |  |  |  |  |  |  |
| Levantamento do problema | **X** |  |  |  |  |  |  |  |  |
| Revisão Bibliográfica | **X** |  |  |  |  |  |  |  |  |
| Justificativa | **X** |  |  |  |  |  |  |  |  |
| Leitura de artigos | **X** |  |  |  |  |  |  |  |  |
| Leitura de Livros | **X** |  |  |  |  |  |  |  |  |

**Fonte:** Elaborado pelo autor

1. **Orçamento**

Para realização desse projeto houve gastos conforme a Tabela 5.

**Tabela 5**: Orçamentos pra realização do Projeto.

|  |  |
| --- | --- |
| **GASTOS** | **VALOR (R$)** |
|  |  |
|  |  |
| **TOTAL** |  |

**Fonte:** Elaborado pelo autor.

Obs**.:** Os custos do projeto serão pagos pelo acadêmico.

1. **Referências**

CARVALHO, Lucas August. **O que é jQuery?** <Disponível em: http://www.webulando.com.br/jquery/o-que-e-jquery>. Data de acesso: 17 mar. 2015

MOTTO, Todd. **Ultimate guide to learning AngularJS in one day.** Disponível em: <http://toddmotto.com/ultimate-guide-to-learning-angular-js-in-one-day/> Acesso em: 17 mar. 2015.

PLANKY, Ricardo. **Definição, restrições e benefícios do modelo de arquitetura REST**, 2014. <Disponível em: c>. Data de acesso: 16 mar. 2015

SOUZA, Cezar Junior. **Zend Framework 1.11**. Disponível em: < http://pt.slideshare.net/cezar08/zend-framework-111>. Data de acesso: 16 mar. 2015

STENZEL, Carlos. **JavaScript um pouco de história**, 2013. Disponível em: < http://www.carlosstenzel.com/TI/javascript-um-pouco-de-historia/>. Data de acesso: 18 mar. 2015

ALBUQUERQUE, Toni. **Paradigmas de programação**, 2010. Disponível em: < http://www.profissionaisti.com.br/2010/10/paradigmas-de-programacao/>. Data de acesso: 18 mar. 2015

SATO, Fernando. **Programação estruturada**, 2009. Disponível em: < http://www.fisica.ufjf.br/~sjfsato/fiscomp1/node23.html>. Data de acesso: 18 mar. 2015

PRADO, Romilson. **História do JavaScript**, 2014.Disponível em: < http://www.romilsonprado.com.br/historia-do-javascript/>. Data de acesso: 18 mar. 2015

MEDEIROS, Higor. **Introdução ao Padrão MVC.** Disponível em: < http://www.devmedia.com.br/introducao-ao-padrao-mvc/29308 >. Data de acesso: 19 mar. 2015

BASTOS, Daniel Flores. **O que é Model-view-controller (MVC) ?**, 2011. Disponível em: < http://www.oficinadanet.com.br/artigo/desenvolvimento/o\_que\_e\_model-view-controller\_mvc >. Data de acesso: 23 mar. 2015

HECKERT, Aurélio Aurium. **O que é SVG ?**, 2008.Disponível em: < http://wiki.softwarelivre.org/InkscapeBrasil/SVG>. Data de acesso: 23 mar. 2015

FRANKLIN, Alysson. **Tenha o DOM.** Entenda o que é o Document Object Model e tenha o DOM, 2011.Disponível em: < http://tableless.com.br/tenha-o-dom/>. Data de acesso: 23 mar. 2015

CHASE, Nicholas. **Entendendo a Zend Framework, Parte 1: O Básico**, 2006. Disponível em: < http://www.ibm.com/developerworks/br/library/os-php-zend1/>. Data de acesso: 24 mar. 2015

SILVA, Paulo. **Seja organizado, sempre!**, 2014.Disponível em: < https://medium.com/@Neocite/seja-organizado-sempre-5464845d328e>. Data de acesso: 24 mar. 2015

JUNQUEIRA, Alvaro R. B., COSTA, André Fernandes, LIRA, Édson Carlos. **Design Patterns: Conceitos e Aplicações**, 1998.Disponível em: < http://www.dcc.ufrj.br/~schneide/PSI\_981/gp\_6/design\_patterns.html>. Data de acesso: 26 mar. 2015

HUMBERTO, Caio. **Design Patterns – Introdução – Parte 1**, 2010.Disponível em: < http://www.devmedia.com.br/design-patterns-introducao-parte-1/16780>. Data de acesso: 26 mar. 2015

CARVALHO, Alan. **JavaScript.** Rio de Janeiro: Book Express Ltda., 2001. 228 p.

SILVA, Maurício Samy. **JavaScript:** Guia do programador. São Paulo: Novatec Editora Ltda., 2010. 597 p.

SILVA, Maurício Samy. **JQuery:** A Biblioteca do Programador JavaScript. 3ª Edição. São Paulo: Novatec Editora Ltda., 2013. 540 p.

W3SCHOOLS. **XML DOM Tutorial**. Disponível em: < http://www.w3schools.com/dom/>. Data de acesso: 01 abr. 2015

JQUERY. **Downloading jQuery.** Disponível em:< http://jquery.com/download/>. Data de acesso: 06 abr. 2015.

NETBEANS. **NetBeans IDE - A maneira mais inteligente e rápida para Código.** Disponível em: < https://netbeans.org/features/index.html>. Data de acesso: 07 abr. 2015.