RELATÓRIO FINAL

PARTE I

Título do plano de trabalho do bolsista	Uma Ferramenta para Conversão de Código JavaScript Orientado a Objetos em ECMA 5 para ECMA 6
Título do projeto do orientador	Uma Ferramenta para Conversão de Código JavaScript Orientado a Objetos em ECMA 5 para ECMA 6
Nome do bolsista	Matheus Russignoli do Nascimento
Nome do orientador	César Francisco de Moura Couto
Grupo de pesquisa	Linguagens e Ambientes de Programação - UFMG
Período de vigência	01/03/2017 a 28/02/2018

RESUMO (Máximo 220 palavras)

ECMAScript, em sua nova versão, ECMAScript 6, introduz uma nova sintaxe para programação orientada a objetos. A proposta é manter o funcionamento baseado em protótipos, mas com uma forma de implementação mais conveniente e sintaticamente próxima dos conceitos tradicionais de orientação a objetos. Para que os desenvolvedores possam usufruir dos benefícios desta nova versão, este trabalho de pesquisa apresenta uma ferramenta, intitulada *ecms5to6* para migração de classes de código em ECMAScript 5 para nova sintaxe proposta em ECMAScript 6. A ferramenta proposta é responsável por converter estruturas de código orientado a objetos usando *prototypes* implementados em ECMAScript 5 para estruturas de código orientada a objetos em ECMAScript 6, tais como `class`, `constructor` e `extends`. Adicionalmente, este trabalho de pesquisa reporta um estudo de caso com 10 softwares reais e populares implementados em JavaScript. Este estudo de caso foi usado para avaliar a eficácia e eficiência da ferramenta *ecms5to6* em realizar a conversão de estruturas de código para ECMAScript 6. Foi concluído que em todos 10 softwares analisados, a ferramenta proposta foi capaz de realizar a conversão de código para ECMAScript 6.

1. INTRODUÇÃO

JavaScript é uma linguagem de programação inicialmente projetada em meados da década de 90 para estender páginas web com código executável. Desde então, sua popularidade e relevância tem crescido consideravelmente [1, 2]. Por exemplo, JavaScript é a linguagem mais popular no GitHub (um repositório de software público), considerando novos repositórios criados por linguagem. É também reportado que a linguagem é usada por 97 de todos os 100 mais populares sites da web [3]. Concomitantemente com o seu aumento de popularidade, o tamanho e a complexidade de softwares JavaScript está em constante crescimento. A linguagem é usada para implementar diversos tipos de aplicações, tais como clientes de email, aplicações de escritórios, IDEs, etc, as quais podem alcançar centenas de milhares de linhas de código[1].

Desde sua criação, programas em JavaScript tem sido implementados seguindo diversos paradigmas, como por exemplo os paradigmas de programação imperativa e funcional. Alem destes, o paradigma da programação orientada a objetos é bastante utilizado [4], usando princípios de prototipagem [5]. Este conceito de prototipagem define uma abordagem diferente da orientação a objetos tradicionalmente utilizada em linguagens como Java ou C#. Adicionalmente, existe um problema sintático, dado que não existem em JavaScript palavras reservadas comumente utilizadas, como por exemplo, `class`. Essas diferenças podem acabar causando dificuldade na implementação do código orientado a objetos.

[1] http://sohommajumder.wordpress.com/2013/06/05/gmail-has-biggest-collection-of-javascript-code-lines-in-the-world, verificado em 11/02/2015

Essa divergência sintática foi alvo das atualizações da ECMAScript, que em sua nova versão, ES 6 (ECMAScript 6) [6] introduz uma nova sintaxe para classes. A proposta é manter o funcionamento baseado em protótipos [7], mas com uma forma de implementação mais conveniente e sintaticamente próxima dos conceitos tradicionais de orientação a objetos. Para que os desenvolvedores possam usufruir dos benefícios desta nova versão, este trabalho de pesquisa apresenta uma ferramenta, intitulada *ecms5to6* para migração de classes de código JavaScript ES 5 (ECMAScript 5) [8] para nova sintaxe proposta de ES 6. Com isso, espera-se ajudar os milhares de desenvolvedores que possuem código que emula classes de acordo com a sintaxe antiga a se beneficiarem de forma automática da sintaxe nativa de classes proposta pela ES 6.

A ferramenta *ecms5to6* é *open-source* e foi implementada utilizando a linguagem de programação Java. Sua principal função é permitir desenvolvedores de software Javascript converter código JavaScript orientado a objeto em ES 5 para ES 6. Mais especificamente, a ferramenta proposta é responsável por converter estruturas de código orientado a objetos usando *prototypes* implementados em JavaScript ES 5 para estruturas de código orientada a objetos em JavaScript ES 6, tais como `class`, `constructor` e `extends`.

Adicionalmente, este trabalho de pesquisa reporta um estudo de caso com 10 softwares reais e populares implementados em JavaScript. Este estudo de caso foi usado para avaliar a eficácia e eficiência da ferramenta *ecms5to6* em realizar a conversão de estruturas de código JavaScript ES 5 para estruturas de código JavaScript ES 6. Foi concluído que em todos 10 softwares analisados, a ferramenta proposta foi capaz de realizar a conversão de código ES 5 para ES 6.

2. MATERIAIS E MÉTODOS

Conforme ilustrado na Figura 1, a ferramenta *ecms5to6* é composta de dois módulos: detecção de classes já implementado em~[9] e conversão para ES 6. O módulo de detecção de classes é responsável por ler arquivos JavaScript (JS), gerar arquivos JSON contendo a árvore de sintaxe abstrata e detectar classes, métodos, atributos e relacionamentos de herança entre classes a partir da árvore de sintaxe abstrata. O módulo de conversão para ES 6 é responsável por converter estruturas de código orientado a objetos usando *prototypes* implementados em JavaScript ES 5 para estruturas de código orientada a objetos específicas para JavaScript ES 6, tais como `class`, `constructor` e `extends`.

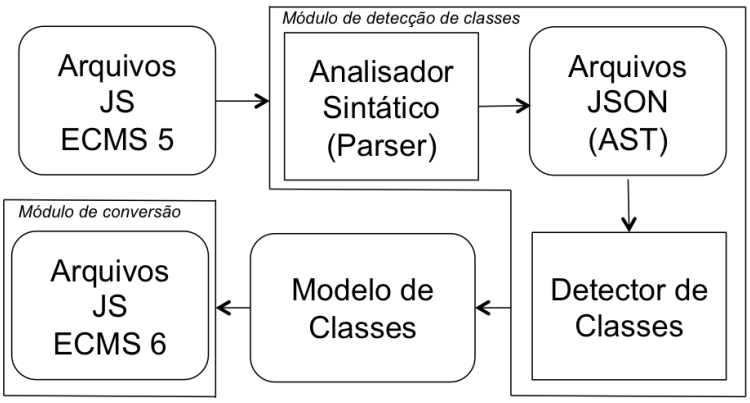

Figura 1 - Arquitetura da ferramenta *ecms5to6*

Módulo de detecção de classes: este módulo recebe como entrada arquivos JavaScript (JS) representando o código fonte da aplicação de origem a ser analisada. A partir dos arquivos JS, este módulo gera a árvore de sintaxe abstrata (AST) no formato JSON utilizando a biblioteca Esprima--um parser JavaScript. Em seguida, este módulo persiste as ASTs geradas em arquivos JSON. A partir dos arquivos JSON, este módulo aplica a estratégia descrita em [9] para detectar classes e gera objetos Java que representam classes contendo atributos, métodos e relacionamentos de herança.

Módulo de conversão em ES 6: este módulo receberá como entrada uma lista de objetos representando classes provenientes do módulo de detecção de classes. Esta lista de objetos compõe o modelo de classes. Em seguida, este módulo implementará um algoritmo de conversão de código que mapea o modelo de classes para estruturas de código específicas de programação orientada a objetos em JavaScript ES 6.

3. RESULTADOS

Um estudo de caso com sistemas de software reais implementados em JavaScript foi realizado para avaliar se a ferramenta *ecms5to6* é efetivamente capaz de converter estruturas de código JavaScript ES 5 para estruturas de código JavaScript ES 6. O contexto deste estudo consiste de 10 softwares JavaScript populares e reais disponíveis no GitHub.

Basicamente, este estudo objetivou responder a seguinte questão de pesquisa (QP):

QP: A ferramenta *ecms5to6* é capaz de converter estruturas de código JavaScript ES 5 para estruturas de código JavaScript ES 6?

A Figura 2 apresenta as diferenças na codificação do arquivo `/algorithms.js/data_structures/bst.js` pertencente ao software *algorithms.js* antes e após a execução da ferramenta *ecms5to6*. Como pode ser observado, a função `Node` foi classificada como uma classe pela ferramenta. Consequentemente, a ferramenta converteu a função `Node` para a nova sintaxe de classe especificada em JavaScript ES 6 ao adicionar as palavras reservadas `class` e `constructor` nas linhas 46 e 47 do arquivo `bst_6.js`.

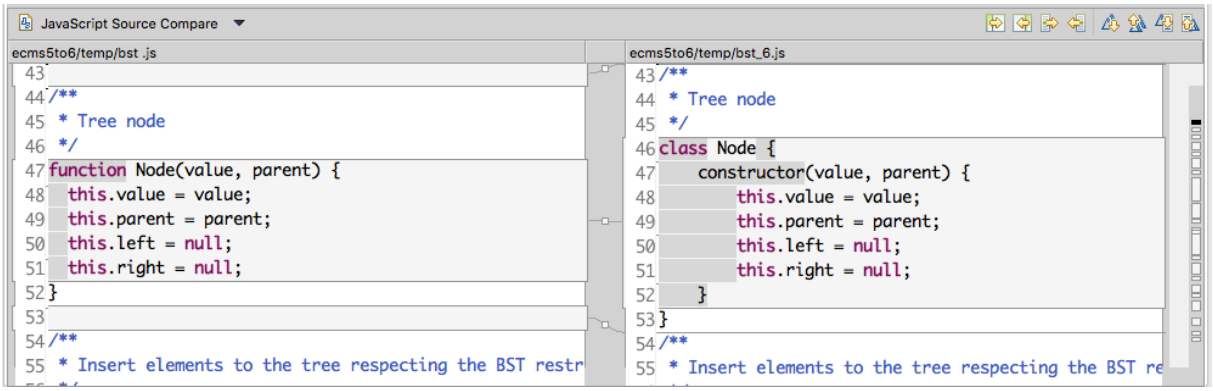

Figura 2 - Diferenças na codificação do arquivo `bst.js` antes e após a execução da ferramenta.

A Figura 3 apresenta as modificações que foram realizadas no arquivo `/algorithms.js/data_structures/heap.js`. Como pode ser observado, a função `MinHeap` foi identificada e classificada como classe. A ferramenta converteu a função para a sintaxe em JavaScript ES 6 assim como na anterior Figura 2. Mais especificamente, foram adicionadas as palavras `class` e `constructor`. Entretanto nessa conversão pode-se notar o aparecimento da palavra reserva `super`. `Super` é uma palavra reservada para invocar o método construtor da classe pai.

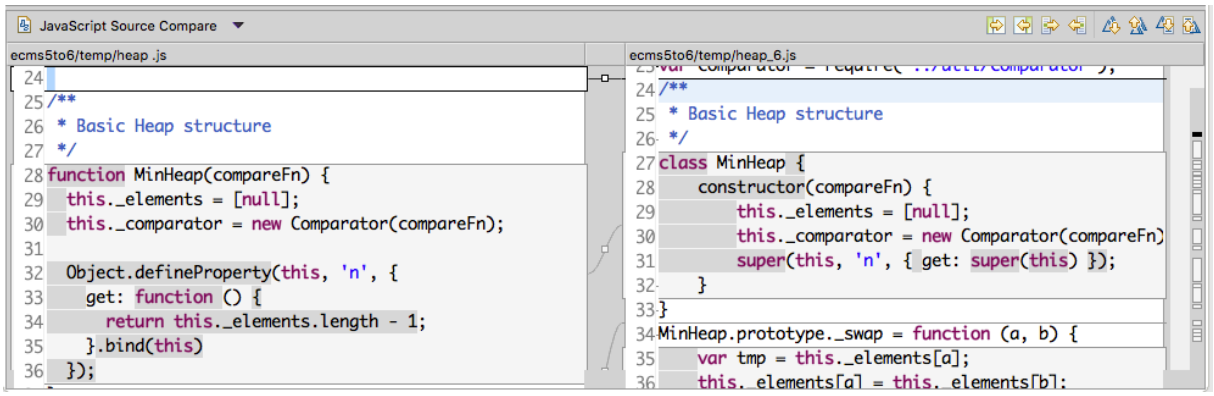

Figura 3 - Diferenças na codificação do arquivo `heap.js` antes e após a execução da ferramenta.

A Figura 4 apresenta a refatoração do arquivo `/mustache.js/mustache.js/Scanner.js` pertencente ao software *mustache.js*. Como pode ser observado, a ferramenta converteu para nova sintaxe em JavaScript ES 6. É possível observar também que o arquivo resultante teve uma redução em sua quantidade de linhas. Além da conversão, a ferramenta é capaz de remover linhas desnecessárias, como pode ser visto nas linhas 271 e 281 do aquivo `ecms5to6/temp/mustache.js`.

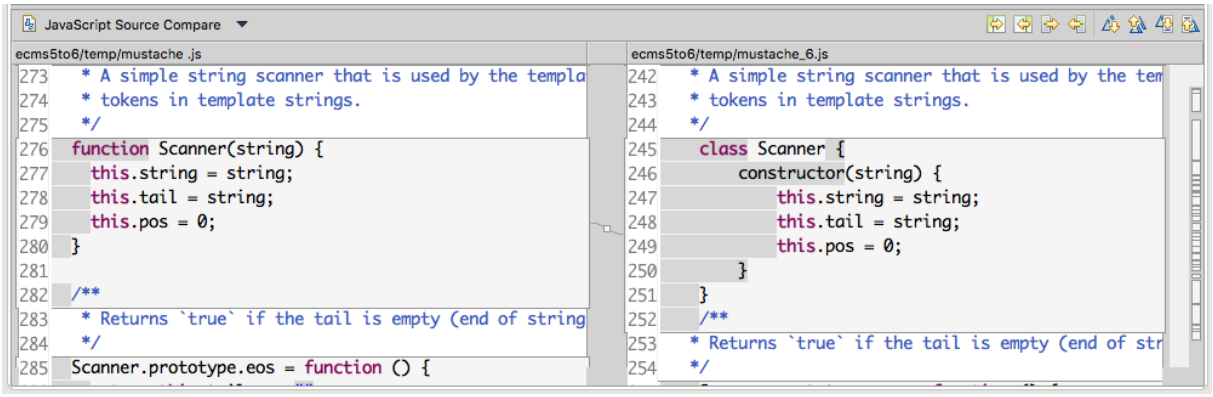

Figura 4 - Diferenças na codificação do arquivo `mustache.js` antes e após a execução da ferramenta.

A Figura 5 apresenta o arquivo `/pdf.js/src/core/Pattern.js` pertencente ao software *pdf.js*. Nesse exemplo, a ferramenta foi capaz de identificar e refatorar a função `MeshStreamReader` que é classificada como classe em JavaScript ES 6 aninhada a função `MeshClosure`.

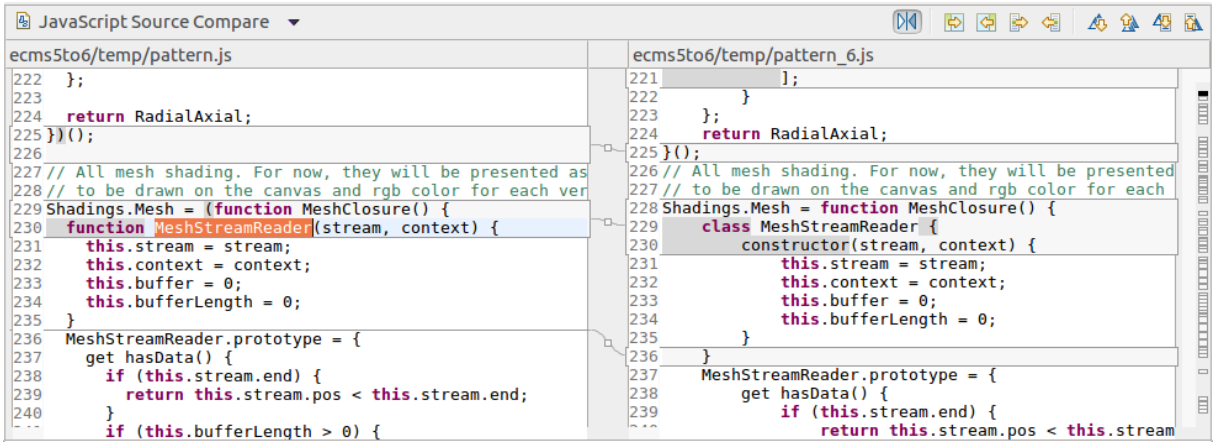

Figura 5 - Diferenças na codificação do arquivo `pattern.js` antes e após a execução da ferramenta.

A Figura 6 apresenta a refatoração do arquivo `/backbone/backbone.js/surrogate.js` pertencente ao software *backbone.js*. A Ferramenta foi capaz de converter a estrutura de código de JavaScript ES 5 para JavaScript ES 6. Nessa figura, pode-se observar na linha 1653 do arquivo `ecms5to6/temp/backbone.js` que desenvolvedor declarou uma *Function Expression* ao utilizar a variável `Surrogate` e atribuir uma função a ela. Mesmo neste caso, a ferramenta *ecms5to6* foi capaz de realizar a conversão. O trecho de código entre as linhas 1630 e 1634 do arquivo `ecms5to6/temp/backbone_6.js` apresenta o código convertido e indentado.

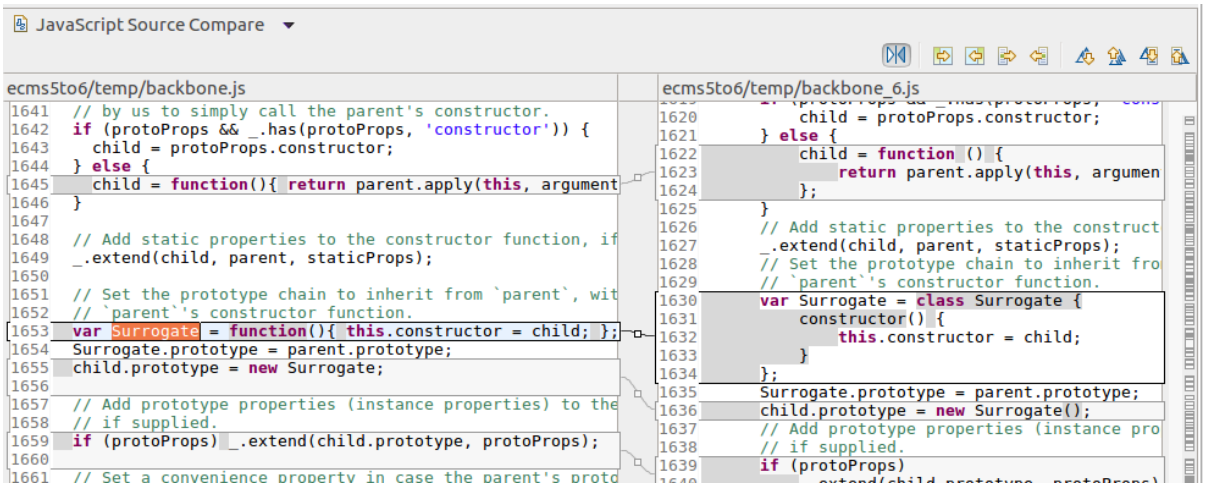

Figura 6 - Diferenças na codificação do arquivo `backbone.js` antes e após a execução da ferramenta.

A partir das ilustrações apresentadas nas figuras anteriores, pode-se concluir que a resposta para a QP é que a ferramenta é completamente capaz de converter estruturas de código JavaScript ES 5 para estruturas de código JavaScript ES 6 em software reais. A ferramenta se encontra publicamente disponível em:

```
https://github.com/cesarfmc/ecms5to6
```

4. DISCUSSÃO

A discussão dos resultados foi descrita detalhadamente na Seção 3.

5. REFERÊNCIA BIBLIOGRÁFICA

[1] H. Kienle. It's about time to take javascript (more) seriously. *IEEE Software*, 27(3):60–92, 2010.

[2] A. Nederlof, A. Mesbah, and A. van Deursen. Software engineering for the web: the state of the practice. In *36th International Conference on Software Engineering (ICSE)*, pages 4–13, 2014.

[3] G. Richards, S. Lebresne, B. Burg, and J. Vitek. An analysis of the dy- namic behavior of JavaScript programs. In *Conference on Programming Language Design and Implementation (PLDI)*, pages 1–12, 2010.

[4] Leonardo Silva, Miguel Ramos, Marco Tulio Valente, Nicolas Anquetil, Alexandre Bergel. Does Javascript Software Embrace Classes?, In *22nd International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 1-10, 2015.

[5] Marcus Arnstrom, Mikael Christiansen, and Daniel Sehl- berg. Prototype-based programming, 2003 (acessado em 21 de novembro de 2016). http://www.idt.mdh.se/kurser/-cd5130/msl/2003lp4/reports/prototypebased.pdf.

[6] European association for standardizing information and communication systems (ECMA). ECMA-262: ECMAScript language specification. 6.0 edition, 2015.

[7] Gunther Blaschek. Object-Oriented Programming: with Prototypes. 1th edition edition, 2012.

[8] European association for standardizing information and communication systems (ECMA). ECMA-262: ECMAScript language specification. 5.1 edition, 2011.

[9] Mauro Mendonça and César Couto. Uma ferramenta para detectar e inspecionar classes em JavaScript. In V Seminário de Iniciação Científica do Instituto Federal do Triângulo Mineiro (SIN), pages 1–1, 2015.

PARTE II

1. ATIVIDADES REALIZADAS NO PERÍODO

- Testes no módulo de detecção de classes em sistemas reais.
- Desenvolvimento de uma aplicação teste em Java para exercitar a programação orientada a objetos.
- Estudo da sintaxe POO de JavaScript ES 6.
- Estudo da ferramenta Esprima, um parser JavaScript responsável por gerar a árvore de sintaxe abstrata (AST) no formato JSON.
- Estudo da AST no formato JSON.
- Estudo de algoritmos e técnicas de conversão de código.
- Implementação do módulo de conversão.
- Realização de um estudo de caso com 10 softwares populares em JavaScript para avaliar a eficácia e eficiência da ferramenta *ecms5to6.*

2. PARTICIPAÇÃO EM EVENTOS CIENTÍFICOS E PUBLICAÇÕES

[1] Giullian Barbosa, César Couto. "Uma Ferramenta para Conversão de Código JavaScript Orientado a Objetos em ECMA 5 para ECMA 6". In VII Seminário de Iniciação Científica do Instituto Federal do Triângulo Mineiro (SIN), 2017.

3. ANEXOS

VII Seminário de Iniciação Científica e Inovação Tecnológica do IFTM – VII SIN-IFTM
Uberaba, MG, 8 de junho de 2017

UMA FERRAMENTA PARA CONVERSÁO DE CÓDIGO JAVASCRIPT ORIENTADO A OBJETOS EM ECMA 5 PARA ECMA 6

Giullian Faria Barbosa[1]; **César Francisco de Moura Couto**[2]

JavaScript é uma linguagem de programação inicialmente projetada em meados da década de 90 para estender páginas web com código executável. Desde então, sua popularidade e relevância tem crescido consideravelmente. Por exemplo, JavaScript é a linguagem mais popular no GitHub (um repositório de software público), considerando novos repositórios criados por linguagem. É também reportado que a linguagem é usada por 97 de todos os 100 mais populares sites da web. Concomitantemente com o seu aumento de popularidade, o tamanho e a complexidade de softwares JavaScript está em constante crescimento. A linguagem é usada para implementar diversos tipos de aplicações, tais como clientes de email, aplicações de escritórios, IDEs, etc, as quais podem alcançar centenas de milhares de linhas de código . Desde sua criação, programas em JavaScript tem sido implementados seguindo diversos paradigmas, como por exemplo os paradigmas de programação imperativa e funcional. Alem destes, o paradigma da programação orientada a objetos é bastante utilizado, usando princípios de prototipagem. Este conceito de prototipagem define uma abordagem diferente da orientação a objetos tradicionalmente utilizada em linguagens como Java ou C#. Adicionalmente, existe um problema sintático, dado que não existem em JavaScript palavras reservadas comumente utilizadas, como por exemplo, *class*. Essas diferenças podem acabar causando dificuldade na implementação do código orientado a objetos. Essa divergência sintática foi alvo das atualizações da ECMAScript, que em sua nova versão, ES 6 (ECMAScript 6) introduz uma nova sintaxe para classes. A proposta e manter o funcionamento baseado em protótipos, mas com uma forma de implementação mais conveniente e sintaticamente próxima dos conceitos tradicionais de orientação a objetos. Para que os desenvolvedores possam usufruir dos benefícios desta nova versão, este trabalho de pesquisa apresenta uma ferramenta, intitulada *ecms5to6* para migração de classes de código JavaScript ES 5 para nova sintaxe proposta de ES 6. Com isso, espera-se ajudar os milhares de desenvolvedores que possuem código que emula classes de acordo com a sintaxe antiga a se beneficiarem de forma automática da sintaxe nativa de classes proposta pela ES 6.

Palavras-chave: Classes em JavaScript, Parser, ECMAScript 6.

Apoio: Fapemig.

1Estudante, IFTM *Campus* Paracatu, MG, bolsista FAPEMIG. giullianfb@yahoo.com.br
2Professor, IFTM Campus Paracatu, MG, Dr. Ciência da Computação. cesarcouto@iftm.edu.br

Local: Paracatu-MG Data: 12/06/2018.

Assinatura do Coordenador