Descrição sintática e semântica de uma linguagem de programação

Diego P. da Jornada* Milton C. de Oliveira†

23 de março de 2015

Resumo

Este artigo descreve a linguagem *Javascript* demonstrando 10 símbolos não terminais utilizando gramática EBNF.

Introdução

Dentro do escopo da disciplina de *Paradigmas de Linguagens de Programação* o primeiro trabalho pode ser resumido da seguinte maneira: deve-se selecionar uma linguagem de programação e descrever, utilizando gramática EBNF, dez símbolos não terminais da linguagem.

Este artigo trata da linguagem de programação *Javascript* que é utilizada para que os *browsers* executem scripts no lado do cliente.

1 História

JavaScript, em 1995, foi desenvolvido por Brendan Eich da NetScape com o nome de Mocha, porém o nome mudou para LiveScript e por fim JavaScript. A escolha final do nome causou confusão dando a impressão de que a linguagem foi baseada em java, sendo que tal escolha foi caracterizada por muitos como uma estratégia de marketing da Netscape para aproveitar a popularidade do recém-lançado Java. Javascript é uma linguagem normalmente usada para que os *browsers* possam executar scripts na máquina cliente; com isso, por exemplo, uma página *web* pode ter seu conteúdo modificado utilizando estes scripts.

*diego.jornada@acad.pucrs.br
†milton.oliveira@acad.pucrs.br

Javascript é um dialeto de ECMAScript e considerada uma linguagem multi-paradigma, tendo funcionalidades os seguintes paradigmas: Imperativo, Orientado a objeto e Funcional.

JavaScript é usado para criar conteúdo web dinâmico e interativo. Tem várias aplicações, incluindo compras pela internet e redes de publicidade.

2 Acessibilidade

Assumindo que o usuário não tenha desabilitado sua execução, pode-se utilizar JavaScript do lado cliente para melhorar a experiência de um usuário com deficiência física ou visual.

Leitores de tela utilizados por pessoas cegas ou com visão parcial podem detectar a presença de JavaScript e dessa forma acessar e ler o DOM da página depois que seus scripts foram executados. Nestes casos recomenda-se que o HTML seja o mais conciso, navegável e rico semanticamente possível, tendo a página scripts ou não. Não se recomenda que o código JavaScript de uma página seja totalmente dependente do eventos provenientes do mouse já que usuários que não conseguem ou optam por não usar o mouse não estarão aptos a colher os benefícios de tal código. Da mesma forma, embora hyperlinks e webforms possam ser navegados e operados do teclado, JavaScript voltado para acessibilidade não deve requerer um teclado para acessar tais eventos. JavaScript conta com eventos independentes do dispositivo de usuário tais como onfocus e onchange que são mais recomendados na maioria dos casos.

Não se recomenda utilizar JavaScript de um jeito que seja confuso ou desorientador para qualquer usuário da internet. Por exemplo, usar JavaScript para alterar ou desabilitar a funcionalidade normal de um navegador, tal como mudar a forma com que o botão direito ou o evento de atualização funcionam, deve ser evitado. Da mesma forma, eventos de interrupção que o usuário pode não estar ciente reduzem a sensação de controle do usuário, assim como scripts inesperados que mudam o conteúdo da página.

Frequentemente o processo de fazer páginas web complexas tão acessíveis quanto possível se transforma em um problema não trivial, onde certas decisões são assunto de debate e opinião. Entretanto, tecnologias assistivas estão constantemente evoluindo e novas recomendações e informações relevantes vem sendo continuamente publicadas na web.

3 Regras Sintáticas

Nesta seção serão descritas as regras sintáticas e, na próxima, as regras semânticas da linguagem(**??**).

Function ::= function [<Name>] (<Params>*) <FunctionBody>

Params ::= <Name> , <Params> | <Name>emph

FunctionBody ::= <Var>* <Statements>*

Statements ::= <Exp> | <Disruptive> | <If> | <Loop>

Loop ::= <While> | <For> | <Do>

If ::= if (<Exp>) <Block> [<Else>]

Else ::= else [<If>] <Block>

While::= while (<Exp>) <Block>

Do ::= do <Block> while (<Exp>);

Block ::= <Statements>

Var ::= var <Name> [= <Expression>];

4 Regras Semânticas

O comando *Function* é usado para definir uma função que executará o *FunctionBody* utilzando parâmetros, se houver, definidos em *Params*.

O comando *Var* é utilizado para a definição de variáveis, sendo estas identificadas por um nome, e caso houver necessidade pode ser atribuído um valor, que é construído a partir de uma *Exp*, que por sua vez é utilizado para que seja definido um valor de forma direta, podendo ser número, string ou booleano. *Exp* também pode ser usada no comando *If* que avalia uma *Exp* e determina se *Block* será executado.

Else é um comando que quando usado após um *If*, que caso a *Exp* a ser avaliada não seja verdadeira o *Block* do *Else* será executado.

Em *Javascript* existem comandos de repetição, *While* verifica se *Exp* é verdadeira e então executa *Block*, enquanto *Do* verifica se *Exp* continua verdadeira após a execução de *Block*.

5 Conclusão

Javascript é uma linguagem importante, pois pode ser usado para facilitar a usabilidade e a interatividade de um determinada página. Já para desenvolvedores *Javascript* por ter elementos do paradigma funcional se mostra muito expressiva possibilitando que sejam escritos códigos menos densos. Por ser multi-paradigma é um linguagem que permite a construção na forma em que o desenvolvedor mais se adapta.