06

Gabarito - PARTE 5 - revisão function declaration vs function expression

Transcrição

Excelente, terminamos nosso sprite. Todavia, mudarei ligeiramente o código. Todas as funções que criamos foram criadas através de function declaration. Isto é, usamos a palavra reservada `function` seguida do nome da função. Há outra forma de definirmos funções através de **function expression*. Essa forma consiste em atribuir para a variável o corpo de uma função. Alterando todo o nosso código temos:

```
var createSprite = function(selector) {

    var hasNext = function() {

        return current + 1 <= last;
    }

    var moveFrame = function(from, to) {

        $el.removeClass(from)
            .addClass(to);
    }

    var nextFrame = function() {

        if (hasNext()) moveFrame (frames[current], frames[++current]);
    }

    var $el = $(selector);

    var frames = [
        'frame1', 'frame2', 'frame3', 'frame4', 'frame5',
        'frame6', 'frame7', 'frame8', 'frame9'
    ];

    var current = 0;

    var last = frames.length - 1;

    $el.addClass(frames[current]);

    return { nextFrame: nextFrame };
}
```

A troca de *function declaration* para *function expression* foi uma questão de gosto deste instrutor. Todavia, há uma ligeira diferente entre as duas abordagem que revisarei aqui.

Quando criarmos uma função através de `function declaration`, mesmo que tenhamos definido a função no meio ou no final do nosso script elas serão sempre içadas para o topo do script. Se declaramos uma função dentro da outra, elas serão içadas para o início da função. Esse procedimento se chamada *function hoisting* (içamento de função). Vejamos um exemplo menos:

```
exibeNome();

function exibeNome() {

    alert('Flávio Almeida');
}
```

O código acima funciona como esperado, porque antes que o interpretador execute nosso código, ele modificará a sua estrutura para:

```
function exibeNome() {

    alert('Flávio Almeida');
}

exibeNome();
```

Todavia, se mudarmos a declaração do nosso código para *function expression* ele não funcionará:

```
exibeNome(); // não funcionará, exibeNome é undefined

var exibeNome = function() {

    alert('Flávio Almeida');
}
```

Isso acontece porque o JavaScript iça também para o topo as declarações de variáveis, mas apenas sua declaração, sem qualquer inicialização. Para o interpretador JavaScript seu código ficará estará assim:

```
var exibeNome; // içou a declaração da variável

exibeNome(); // repare que não há qualquer valor

exibeNome = function() {

    alert('Flávio Almeida');
}
```

Com este exemplo, fica claro a diferença entre as duas abordagens. Este instrutor prefere a segunda forma porque força o desenvolvedor a declarar as expressão de função antes do seu uso, o que é mais compreensível. Além disso, essa forma abre espaço para outras melhorias que fogem o escopo do ECMASCRIPT 5 e entram no escopo do ES2015 (ES6). No final do curso teremos um bônus no qual eu migrarei todo código para ES2015 para que o aluno veja como essa atualização do JavaScript pode ajudá-lo a escrever um código ainda melhor, além de indicar quais cursos de outras carreiras o aluno deve procurar para adiquirir esse conhecimento.

Por fim, o `setInterval` proposto para testar nosso sprite funcionará como esperado. Ao final do teste, você pode removê-lo.