

Gabarito - PARTE 4 - a lógica do hasNext()

Transcrição

Isso já é suficiente para podermos ver a exibição de cada sprite a cada meio segundo. Todavia, há um problema ainda. Não podemos exceder a quantidade de frames definida no array. Isso significa que ao toda vez que `nextFrame()` for chamado, ele precisará verificar se ainda há o próximo frame. Havendo, mudamos para o próximo frame, caso contrário, ficamos no frame atual, no caso, o último frame do sprite que apresenta a força completa.

Vamos criar a função `hasNext()` :

```
function createSprite(selector) {

    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]);

    function moveFrame (from, to) {

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

    function hasNext() {

        return current + 1 <= last;
    }

    function nextFrame() {

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

    return { nextFrame: nextFrame };
}
```

Excelente, nosso sprite ao chegar ao último frame fica parado. Por fim, podemos organizar um pouco melhor nosso código movendo para o início do bloco da função `createSprite` as funções utilitárias que criamos:

```
function createSprite(selector) {

    function hasNext() {
```

```
    return current + 1 <= last;
  }

  function moveFrame (from, to) {

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

  function nextFrame() {

    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 };
}
```