05

Gabarito

Transcrição

Dentro da função `processaChute`, declararei três variáveis. A primeira receberá uma expressão regular criada a partir do chute informado pelo jogador. Nessa expressão, utilizo o modificador `gi`. O modificador `g` é para realizar uma pesquisa global, isto é, mesmo que ele faça o match da expressão na primeira letra, a expressão continuará sendo aplicada até o fim do texto, no caso, nosso alvo será `palavraSecreta`. O modificador `i` é para indicar que não estamos levando em consideração na comparação se a letra esta em maiúscula ou minúscula. A variável `resultado` guardará o resultado na nossa expressão regular e por fim `acertou` será um marcador para sabermos se o jogador acertou ou não a letra.

```
var processaChute = function (chute) {

    var exp = new RegExp(chute, 'gi');
    var resultado;
    var acertou = false;
};
```

Declaração de variáveis mais sucinta

Todavia, podemos simplificar a declaração dessas variáveis da seguinte maneira:

```
var processaChute = function (chute) {

    var exp = new RegExp(chute, 'gi')
        , resultado
        , acertou = false;
};
```

Vejam que só precisei declarar `var` uma única vez. Muito bem, agora precisamos utilizar o poder da expressão regular. Não basta sabermos se a letra que o jogador chutou existe ou não em `palavraScreta`, precisamos saber em qual posição ela esta para que possamos atualizar o array de lacunas nesta posição com o chute.

Faremos um teste no console do navegador vermos como utilizaremos a expressão:

```
palavraSecreta = 'calopsita';
exp = new RegExp('a', 'gi');
exp.exec(palavraSecreta) // saída ["a", index: 1, input: "calopsita"]
exp.exec(palavraSecreta) // saída ["a", index: 8, input: "calopsita"]
exp.exec(palavraSecreta) // null, não encontrou mais nada
```

Como podemos ver, a cada chamada de `exp.exec()` recebemos um objeto JavaScript com uma série de propriedades. Uma delas é a posição na qual o chute aparece, ou seja, o `index`. A cada chamada de `exp.exec()` recebemos um novo match até que seja retornado `null` para indicar que a avaliação chegou ao fim.

Com base no que cabamos de constatar no console, vamos utilizar um `while` para automatizar esse processo:

```
var processaChute = function (chute) {

    var exp = new RegExp(chute, 'gi')
        , resultado
        , acertou = false;

    while (resultado = exp.exec(palavraSecreta)) {
        lacunas[resultado.index] = chute;
    }
};
```

Enquando o while estiver sendo executado, é porque há match. Então, em seu bloco, acessamos a posição `lacunas[resultado.index]` e nela atribuímos o chute. Perfeito, pois essa abordagem conseguirá preencher todas as lacunas de um determinado chute. E o erro? Precisamos saber no final do bloco `while` se houve algum erro ou não e, caso haja, precisaremos exibir o próximo frame do sprite da forca. Para isso faremos o seguinte:

```
var processaChute = function (chute) {

    var exp = new RegExp(chute, 'gi')
        , resultado
        , acertou = false;

    while (resultado = exp.exec(palavraSecreta)) {
        acertou = true;
        lacunas[resultado.index] = chute;
    }

    if (!acertou) {
        sprite.nextFrame();
    }
};
```

O valor de `acertou` só será `true` se houver match, caso contrário, continuará como false. Se o usuário não acertou, chamamos `sprite.nexFrame()`.

Todavia, podemos enxugar um pouco mais nosso código com duas mudanças:

```
var processaChute = function (chute) {

    var exp = new RegExp(chute, 'gi')
        , resultado
        , acertou = false;

    while (resultado = exp.exec(palavraSecreta)) {

        acertou = lacunas[resultado.index] = chute;
    }

   if (!acertou) sprite.nextFrame();
};
```

A primeira é a instrução `acertou = lacunas[resultado.index] = chute;`. Nela, em uma tacada só, estamos atribuindo o chute à sua respectiva posição no array `lacunas` e guardamos o valor de `lacunas` na variável acertou. Em JavaScript, qualquer string diferente de branco como " é considerado `true`. Outra mudança que fizemos foi remover o bloco da condição `if`. Aliás, podemos fazer a mesma coisa com o while:

```
var criaJogo = function (sprite) {

    var etapa = 1;
    var lacunas = [];
    var palavraSecreta = '';

    var criaLacunas = function () {

        for (let i = 0; i < palavraSecreta.length; i++) {
            lacunas.push('');
        }
    };

    var proximaEtapa = function () {

        etapa = 2;
    };

    var setPalavraSecreta = function (palavra) {

        palavraSecreta = palavra;
        criaLacunas();
        proximaEtapa();
    };

    var getLacunas = function () {

        return lacunas;
    };

    var getEtapa = function () {

        return etapa;
    };

    var processaChute = function (chute) {

        var exp = new RegExp(chute, 'gi')
            , resultado
            , acertou = false;

        while (resultado = exp.exec(palavraSecreta))
            acertou = lacunas[resultado.index] = chute;

        if (!acertou) sprite.nextFrame();
    };

    return {
        setPalavraSecreta: setPalavraSecreta,
        getLacunas: getLacunas,
        getEtapa: getEtapa,
```

```
            processaChute: processaChute
        }
    };
```