Lista de Exercícios – JavaScript

Professor: Luiz Carlos Felix Carvalho

Jogo da Velha - JavaScript

Nessa lista, vamos reunir o conhecimento já apresentado e construiremos um jogo simples, Jogo da Velha, utilizando, principalmente, JavaScript. Você verá que o quão pouco necessitamos para tal.

Primeiramente, vamos construir o HTML. O jogo da velha tem em sua base uma tabela, com três linhas e três colunas. Em nossa implementação, utilizaremos como base o elemento div e montaremos uma tabela com tal. Segue código HTML:

```
<DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css"/>
        <script type="text/javascript" defer src="script.js"></script>
    </head>
    <body>
        <h1>Jogo da Velha</h1>
        <div id="jogo">
            <div class="linha">
                <div class="casa" id="casa1"></div>
                <div class="casa" id="casa2"></div>
                <div class="casa" id="casa3"></div>
            </div>
            <div class="linha">
                <div class="casa" id="casa4"></div>
                <div class="casa" id="casa5"></div>
                <div class="casa" id="casa6"></div>
            </div>
            <div class="linha">
                <div class="casa" id="casa7"></div>
                <div class="casa" id="casa8"></div>
                <div class="casa" id="casa9"></div>
            </div>
        </div>
        <div id="resultado"></div>
    </body>
</html>
```

Note que já estamos considerando a utilização de um arquivo para script e outro para o CSS. Também já adicionamos nos elementos HTML, as classes CSS que utilizaremos em nosso exemplo. Assim, temos dois div's principais, o de id jogo e o de id resultado. O de id jogo possui uma estrutura interna de div's que será nosso tabuleiro. O div de resultado será utilizado para apresentarmos o vencedor. Cada div que representa uma linha está associado a uma class CSS de nome linha. O mesmo acontece para as colunas, porém, com o nome de casa. Cada casa possui um id único "casa" + identificador. Assim, conseguimos identificar cada casa e, facilmente, conseguimos acessar o conteúdo de cada, o que será utilizado para verificarmos a lógica do jogo.

Sabemos que, o elemento div precisa ser estilizado via CSS para que possamos posicionar um do lado do outro. Para tal, modificaremos seu tamanho e utilizaremos a propriedade float, com o valor left. Assim, posicionaremos e alinharemos os divs a esquerda. Tal forma de organização foi assunto da aula prática 10. Também utilizaremos bordas para distinguir as casas. Assim, segue arquivo CSS:

```css
#jogo{
    width:603px;
    height:600px;
    border:solid 3px
}

.linha{
    height:200px;
    border-bottom:solid 1px;
}

.casa{
    width:200px;
    height:100%;
    border-right:solid 1px;
    float:left;
}
```

Execute e veja o resultado!

Adicionando a lógica do jogo com JavaScript

Vamos iniciar pelo evento que teremos que colocar em cada casa do jogo. A ideia é que ao clicar na casa, o símbolo, xis (x) ou bolinha (o), seja adicionado. Então, teremos que colocar um evento no click de cada casa. Observando o HTML, temos que toda casa possui a classe CSS casa. Vamos, então, acessar todo elemento que possui tal classe. Um método que é ideal para esse acesso é o querySelector, já apresentado anteriormente. Esse método, de document, acessa os elementos por seletor CSS. Ele tem uma variante que retorna todos os elementos encontrados querySelectorAll. Portanto, vamos criar um método que acesse todo elemento com a classe CSS casa e adicione um comportamento no clique em tais elementos:

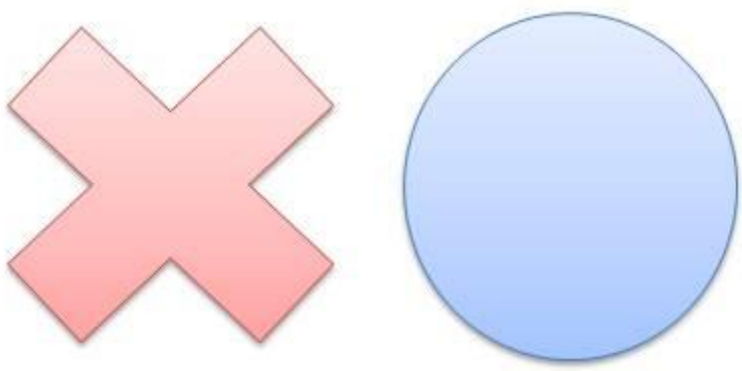
```
function inicioApp() {

    var casas = document.querySelectorAll( ".casa" );

    casas.forEach( element => {

        element.addEventListener( 'click', clickCasa );
    });
}
```

Chamamos o método de inicioApp, pois como vamos colocar um evento em cada casa e tal evento é necessário para jogar, ele deve ser adicionado ao carregar o HTML. O comportamento adicionado é o método clickCasa, que iremos implementar.

O que se deve fazer ao clicar em cada casa? Essencialmente, temos que mostrar um x (xis) ou uma o (bolinha). Mas, como saber qual mostrar? Vamos assumir que os jogadores vão se alternar a cada clique, assim criaremos uma variável global para armazenar tal indicativo de quem jogará.

```
var vez = 1;
```

Também, utilizaremos duas imagens que serão apresentadas indicando qual jogador que clicou na casa x ou o. Para facilitar, vamos dar o nome de 1.png e 0.png nas imagens, de modo que vamos associar diretamente com o conteúdo da variável global 'vez', já definida. Segue imagens:



Porém, não é somente mostrar a imagem. Temos que verificar se a tal casa já não foi marcada por algum jogador. Podemos, fazer da seguinte forma:

1) Acessar o elemento HTML da casa clicada
2) Verificar se ele já foi clicado
3) Se não foi clicado,
    a. colocamos a imagem de tal jogador
    b. alteramos a variável global 'vez' para indicar que a próxima jogada é do outro jogador

c. verificamos se o jogo finalizou, depois de tal jogada.

Assim, temos:

```
var clickCasa = function( event ){

    // acessar elemento clicado
    var element = event.target;

    // obter backgound do elemento
    var bg = element.style.backgroundImage;

    // verificar se o background está vazio
    if( bg == "none" || bg == "" ) {

        //criar uma url para acessar a imagem determinada pela variável vez

        // alterar background do elemento clicado

        // alterar a variável vez (alternando de 1 para 2 ou de 2 para 1)

        // invocar método para verificar fim do jogo (tal método ainda será
        // implementado). Apenas coloque a chamada dele.
    }
}
```

Note que o código apresenta o método do comportamento do clique de cada casa, em que se acessa o elemento clicado através de event.target, depois acessa-se a imagem de background da casa clicada e verifica se há algo nesse background. Se não houver, deve-se marcar a casa com uma opção e fazer as ações de lógica do jogo, comentadas dentro do *if*. Note, também, que a ideia é fazer a exibição da imagem pela propriedade de background-image.

Exercício 1: Implemente o que se pede (nos comentários) no método clickCasa.

Comente a invocação do método de verificação de finalização do jogo, faça o método inicioApp ser invocado quando o HTML é carregado e execute o código.

Obs.: Como fazer o método inicioApp ser invocado quando o HTML é carregado? Uma opção é invocar ele no JavaScript, no fim do script:

```
// Coloque no fim do script
inicioApp();
```

Cada casa já tem seu evento e sabemos que precisamos indicar se o jogo finalizou ou não. Então, vamos implementar!

O método deve ter o nome que você colocou na última linha do último exercício. Aqui, vamos chamar de verificarFimDeJogo(). Para implementá-lo, devemos entender quando o jogo finaliza: quando três casa em linha, coluna ou diagonal estão com o mesmo símbolo. Portanto, antes de implementar verificarFimDeJogo, criaremos um método auxiliar, que vamos chamar de casasIguais, quer verificará se três casas possuem o mesmo símbolo. Esse método deve:

1) Receber três parâmetros, que represente o número (do id, veja o id das casas) de cada casa;
2) Acessar o elemento HTML das três casas;
3) Acessar a propriedade HTML do background-image de cada elemento;
4) Verifique se o background-image do primeiro elemento está em nulo ou vazio (como implementado no método do clique)
    a. Se estiver vazio ou nulo, retorna falso (se qualquer elemento estiver vazio ou nulo, quer dizer que não possui valor e não foi clicado, ou seja, nessas casas informadas não tem regra para finalizar o jogo);
5) Verifique se o background-image das três casas possuem o mesmo valor (se possuírem o mesmo valor, o jogo está finalizado, então devemos):
    a. Verificar se o conteúdo do background possui 1.png ou 2.png (lembre-se que o conteúdo do background-image é uma chamada para a função url, assim é uma string que contém o nome do arquivo da imagem (exemplo: **url("nome-arquivo.jpg")**). Então, deve-se utilizar alguma forma de verificar se na string possui o nome do arquivo; recomenda-se utilizar o método indexOf: varString.indexOf("123"): se em varString contiver o texto "123", retornará um inteiro maior do que -1, se não retornar, não contém).
    b. Para indicar o vencedor, utilizaremos uma variável global 'vencedor'. Então, ao descobrir qual o vencedor (1 ou 2), armazene em tal variável.
    c. Retorne verdadeiro para indicar que o jogo finalizou
6) Caso a condição anterior não indicar a finalização do jogo, retorne falso, indicando que com essas posições o jogo não finalizou.

Abaixo, segue o esquema de tal método:

```
var vencedor = "";

//cada parâmetro é um número que indica uma casa
function casasIguais( a, b, c ){

    // acessar o elemento do parâmetro 1 (id: "casa" + a)
    // acessar o elemento do parâmetro 2 (id: "casa" + b)
    // acessar o elemento do parâmetro 3 (id: "casa" + c)

    // acessar o backgroundImage do elemento de parâmetro 1 (id: "casa" + a)
    // acessar o backgroundImage do elemento de parâmetro 2 (id: "casa" + b)
    // acessar o backgroundImage do elemento de parâmetro 3 (id: "casa" + c)
```

```
    // se o primeiro elemento for nulo (none) ou em branco, retorna falso

    // verificar se o backgroundImage dos três são iguais, se sim tem condição para
    // finalizar o jogo

        // verificar se o conteúdo do backgroundImage contém o arquivo 1

            //se sim, o vencedor é o jogador 1 (armazenar na variável global)

            //se não, o vencedor é o jogador 2 (armazenar na variável global)

        // retornar verdadeiro para indicar que as três casas determinam a
        // finalização o jogo

    // retorna falso, pois com essas três casas o jogo não finaliza

}
```

Exercício 2: Implementar o método que verifica a finalização do jogo recebendo três casas para verificação.

Para finalizar a implementação, precisamos construir o método, já citado, verificarFimDeJogo. Esse método, deve invocar o anterior, casasIguais, passando como parâmetro, a cada invocação, três possibilidades de casas, que se estiverem iguais, finalizam o jogo, por exemplo: 1, 2 e 3; 4, 5 e 6; 7, 8 e 9; etc. São 8 possibilidades. Caso, uma dessas invocações retorne verdadeiro, o jogo deve ser finalizado e o vencedor deve ser informado no div de id resultado. Assim temos:

```
function verificarFimDeJogo(){

    // Verificar se algum trio de casas (que podem finalizar o jogo) estão iguais
            // se sim, acessar o div de id resultado e apresentar uma mensagem que
            //indica o fim do jogo e o número do ganhador.


}
```

Exercício 3: Implementar verificarFimDeJogo().

Para encerrar a execução, é necessário remover o evento das casas, para que se finalize o tabuleiro, mostrando o resultado. Assim, criaremos o método fimApp, que finalizará a execução removendo todos os eventos (listeners):

```
function fimApp() {

    // acessar todos elementos com classe casa
```

```
    // para cada elemento encontrado, remover o listener no 'click', identificado
pelo método clickCasa.
}
```

O método fimApp deve ser invocado logo após o ganhador ser informado. Assim, deve-se alterar o método verificarFimDeJogo, adicionando tal invocação.

Exercício 4: Implementar fimApp() e alteração necessária em verificarFimDeJogo().

Com isto o jogo está finalizado. Divirta-se!

Lista de exercícios:

5) Inclua um botão que reinicialize o jogo, porém exiba o botão somente quando o jogo finalizar. Utilize a propriedade visibility do CSS para que apresentar o elemento (se vazio (“”) ou se com “visible”), ou esconder com o valor “hidden”.
6) A lógica implementada não determina quando o jogo finaliza sem um vencedor. Altere a lógica de forma que, indique quando o jogo finaliza sem um vencedor.