Desenvolvimento - Javascript

Criando um Jogo da Velha em Javascript, HTML e CSS.

Veja neste artigo como desenvolver um Jogo da Velha simples, utilizando apenas recursos básicos das web standards (HTML, CSS e Javascript). O objetivo deste artigo é estimular o estudo destas tecnologias, mostrando que é possível obter resultados, que muitas vezes achamos que são demasiadamente complexos, de forma simples e rápida.

por Joel Rodrigues

O objetivo deste artigo é apresentar um código simples para implementar um “Jogo da Velha”, utilizando apenas recursos básicos das web Standards (HTML, CSS e Javascript). O intuito aqui não é apresentar nenhuma ferramenta ou código de desenvolvimento de jogos avançados, mas sim estimular o estudo das três tecnologias que regem a web, bem como da lógica de programação e da biblioteca jQuery.

Para o jogo, utilizaremos duas imagens quaisquer que, nesse caso, são ilustradas abaixo. O leitor pode fazer uso das figuras de sua preferência, atentando apenas para os nomes, que devem ser mantidos.

Figura 1: (figura do xis)

Figura 2: (figura do círculo)

Todos os arquivos produzidos neste artigo devem ser mantidos na mesma pasta, inclusive as imagens. Caso contrário, o leitor precisará alterara o caminho das referências feitas no código.

Iniciemos então pelo código HTML que estruturará o jogo. Teremos basicamente uma div maior dividida em três linhas e três colunas. O conteúdo da Listagem 1 deve ser salvo com a extensão HTML, no caso, nomeei o arquivo como index.html.

Listagem 1: Arquivo index.html

```
<DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="estilo.css"/>
    <script type="text/javascript" src="http://code.jquery.com/jquery-1.7.2.min.js">
    <script type="text/javascript" src="script.js"></script>
</head>
<body>
        <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>
```

As casas foram numeradas de 1 a 9, da esquerda para a direita, de cima para baixo. Esta identificação é necessária para verificar se a casa está marcada com xis ou com círculo.

Na tag header do HTML foram feitas referências a três arquivos: uma folha de estilos, a biblioteca jQuery e um script próprio deste artigo.

Listagem 2: Arquivo estilo.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;
}
```

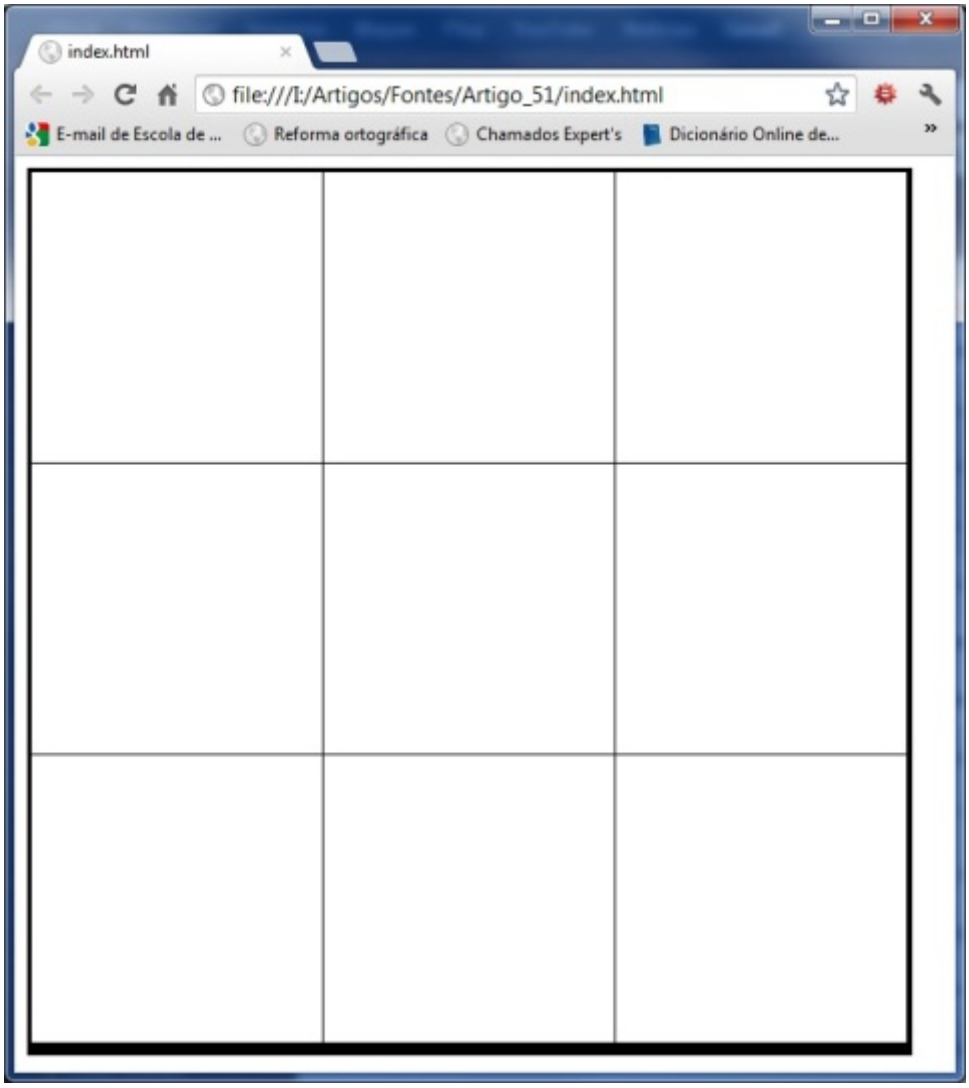

Figura 3: Aparência inicial da página

Vale ressaltar que o design não é o foco deste artigo, mas sim a utilização dos recursos do HTML, CSS e Javascript para a obteção dos resultados desejados.

Listagem 3: Estrutura geral do arquivo script.js

```
$(function(){
//O conteúdo deve ficar aqui
});
```

Então, iniciemos o desenvolvimento do script responsável por “dar vida” ao jogo. Primeiramente, devemos declarar duas variáveis globais que serão utilizadas para identificar de quem é a vez e quem é o vencedor, quando este for definido.

Listagem 4: Declaração de variáveis

```
var vez = 1;
var vencedor = "";
```

Em seguida, implementaremos uma função para verificar se uma fila (linha, coluna ou diagonal) está completamente preenchida por um mesmo jogador. Esta função receberá como parâmetro os índices das três casas a serem verificadas.

Listagem 5: Função para verificar preenchimento de uma fila

```
function casasIguais(a, b, c){
    var casaA = $("#casa"+a);
    var casaB = $("#casa"+b);
    var casaC = $("#casa"+c);
    var bgA = $("#casa"+a).css("background-image");
    var bgB = $("#casa"+b).css("background-image");
    var bgC = $("#casa"+c).css("background-image");
    if( (bgA == bgB) && (bgB == bgC) && (bgA != "none" && bgA != "")){
        if(bgA.indexOf("1.png") >= 0)
            vencedor = "1";
        else
            vencedor = "2";
        return true;
    }
    else{
        return false;
    }
}
```

Caso as três casas verificadas estejam igualmente preenchidas, define-se quem foi o vencedor, tomando como base a imagem com a qual as casas estão marcadas. A variável “vencedor” recebe então o valor “1” ou “2”.

Definiremos agora mais uma função que será responsável por verificar se o jogo acabou, ou seja, utilizando a função casasIguais, verificará se alguma linha, coluna ou diagonal está preenchida e, em caso positivo, exibe uma mensagem informando o vencedor do jogo. Ao final, o evento click das casas é desativado, impedindo a continuação da partida.

Listagem 6: Função para verificar o fim do jogo

```
function verificarFimDeJogo(){
    if( casasIguais(1, 2, 3) || casasIguais(4, 5, 6) || casasIguais(7, 8, 9) ||
        casasIguais(1, 4, 7) || casasIguais(2, 5, 8) || casasIguais(3, 6, 9) ||
        casasIguais(1, 5, 9) || casasIguais(3, 5, 7)
```

```
    }
}
```

Sabemos, porém, que esta função precisa ser chamada de algum ponto do código, de forma a estar constantemente verificando o andamento da partida. Para isso, programaremos o evento click das casas para chamar a função verificarFimDeJogo, afinal, o jogo só termina quando um jogador marcar a terceira casa de uma sequência (ou se nenhum jogador vencer, neste caso, não será exibida nenhuma mensagem).

No código da Listagem 6, quando uma casa é clicada pelo jogador, verificamos o conteúdo do atributo CSS “background-image”, caso o mesmo esteja vazio, preenchemos a casa com a imagem “1.png” ou “2.png”, dependendo da vez. Por fim, alteramos o jogador da vez para que a partida possa prosseguir e invocamos a função verificarFimDeJogo.

Listagem 7: Evento click das casas

```
$(".casa").click(function(){
    var bg = $(this).css("background-image");
    if(bg == "none" || bg == "")
    {
        var fig = "url(" + vez.toString() + ".png)";
        $(this).css("background", fig);
        vez = (vez == 1? 2:1);
        verificarFimDeJogo();
    }
});
```

Caso tenha ficado a dúvida, o conteúdo das listagens 4, 5, 6 e 7 deve ser inserido no arquivo script.js, de acordo com a estrutura mostrada na Listagem 3.

Feito isso, podemos abrir o arquivo index.html no browser (ou atualizar, caso já esteja aberto) e testar o funcionamento do nosso código.

A Figura 4 ilustra ilustra o momento em que o jogador 1 venceu a partida.

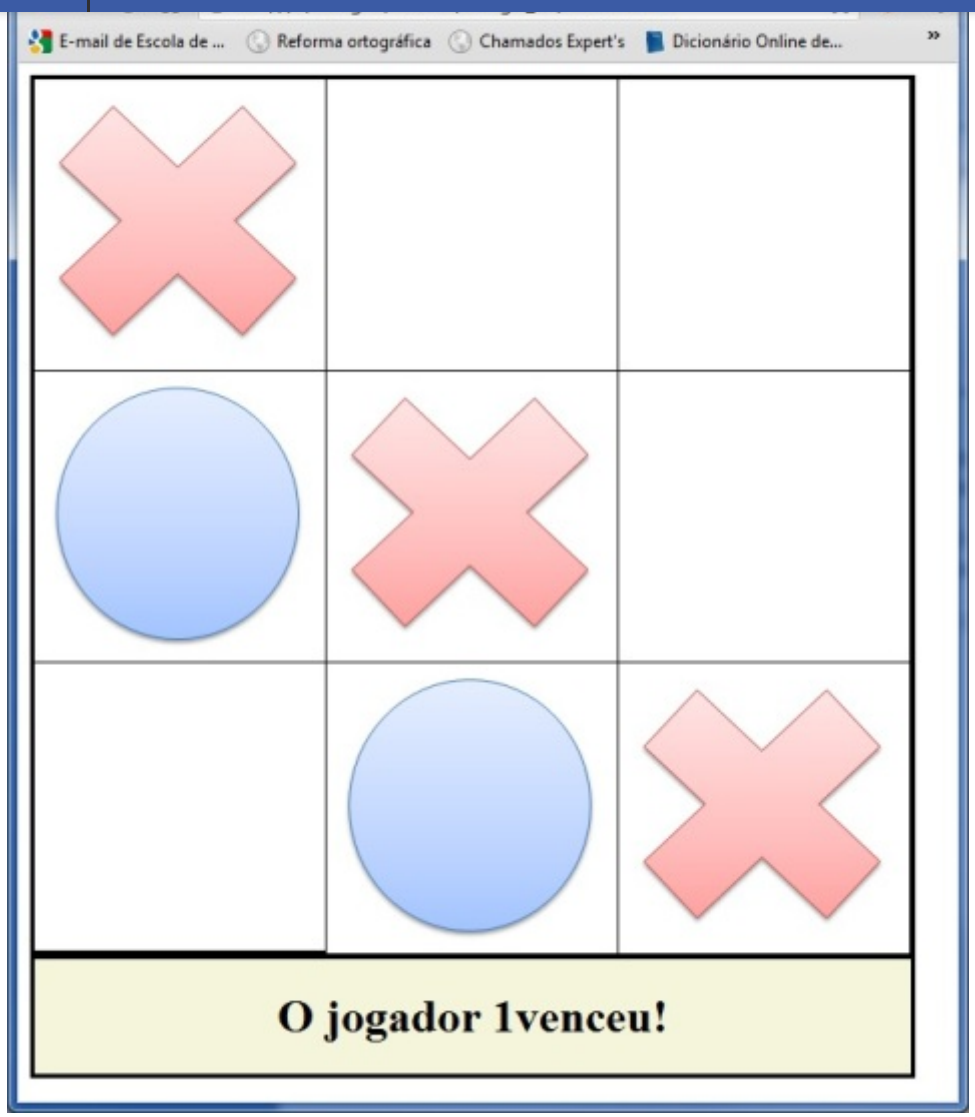

Figura 4: Jogo em funcionamento com vitória do jogador 1

Como vimos, não foram utilizados recursos avançados das web standards, imagens ou editores e mesmo assim, conseguimos desenvolver rapidamente um “Jogo da Velha” simples, porém funcional.

Espero que o conteúdo aqui apresentado possa ser útil no auxílio aos desenvolvedores web e interessados pela área, principalmente aqueles que estão iniciando os estudos dessas tecnologias.

Grato pela atenção, finalizo aqui este artigo. Até a próxima publicação.

Joel Rodrigues - Técnico em Informática - IFRN Cursando Bacharelado em Ciências e Tecnologia - UFRN Programador .NET/C# e Delphi há quase 3 anos, já tendo trabalhado com Webservices, WPF, Windows Phone 7 e ASP.NET, possui ainda conhecimentos em HTML, CSS e Javascript (JQuery).

Javascript

Principais Frameworks de Javascript
Javascript

Conhecendo o HTML5 Notifications API
Javascript

Como inverter links ou textos com Javascript
Javascript

Criando um jogo da velha em DHTML (HTML+Javascript) com jvGame
Javascript