> *Ficha Prática Nº6 (Jogo de Memória – Top 10 e HTML Event Handlers)*

Esta ficha tem como objetivo implementar a gestão do **Top10**, mais propriamente, solicitar ao jogador o respetivo *nickname* e quando a pontuação obtida assim o permitir, entrar na lista dos "Top 10". Esta ficha, conclui a construção do jogo de memória em JS, no contexto das aulas de Linguagens Script.

O resultado desta ficha apresenta-se na figura 1.

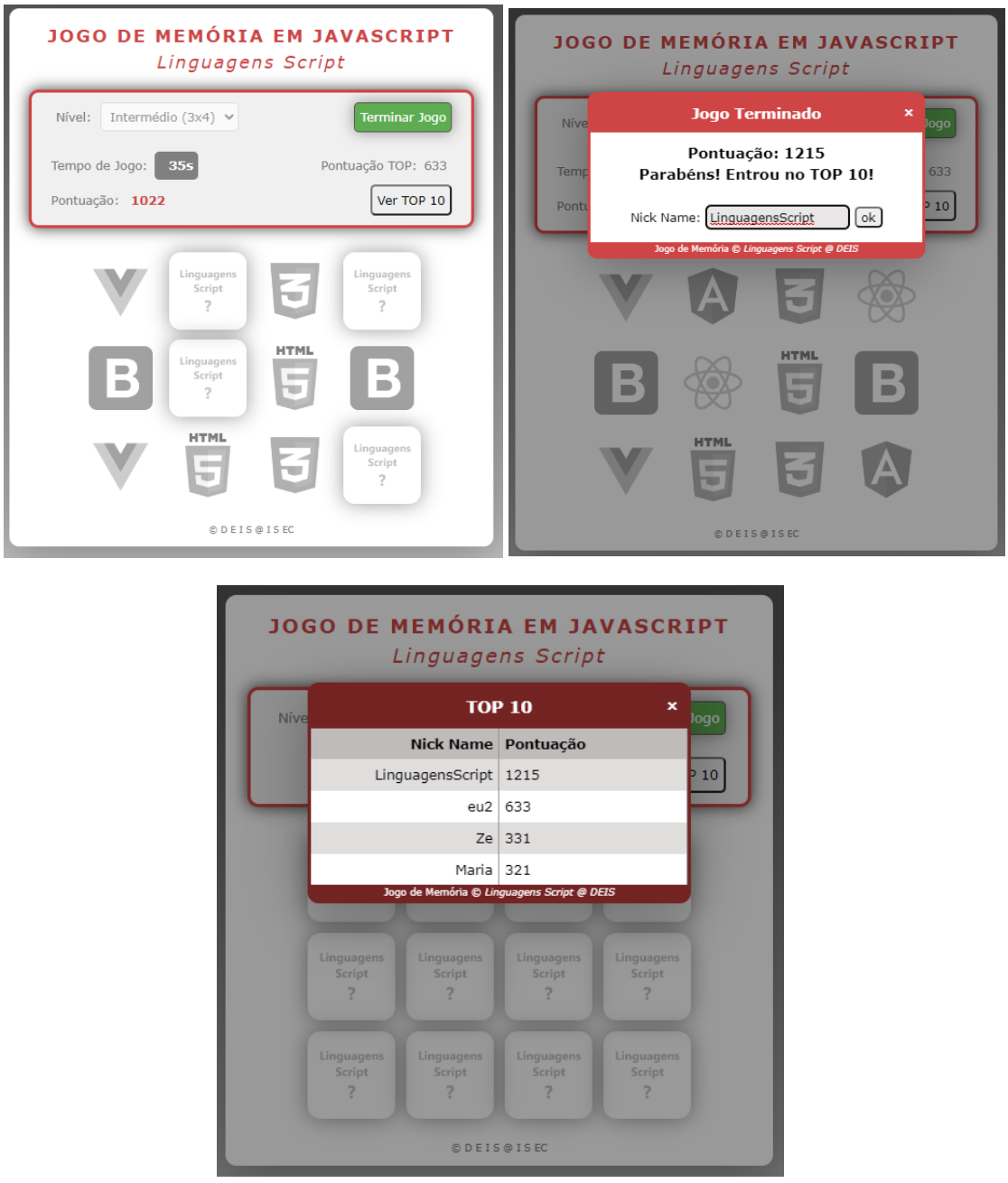

Figura 1 – Jogo de Memória em JavaScript – Imagens da aplicação – Top 10

> Preparação do ambiente

Figura 2 - Jogo (início da ficha)

a. Descompacte o ficheiro **fich6.zip**.

Os alunos que concluíram a resolução da ficha anterior, devem continuar nesse trabalho.

Os restantes alunos podem resolver esta ficha prática, tendo como base o código fornecido juntamente com esta ficha.

b. Inicie o *Visual Studio Code* e abra **a pasta no *workspace*.** Visualize a página **index.html** no browser, no qual terá o aspeto da figura 2.

c. Módulos a implementar:

- **Apresentação do Top10 (Parte I);**
- **Gestão do Top 10 (Parte II);**
- **Persistir dados do Top10 com *localStorage* (Parte III);**
- **HTML Event Handlers (Parte IV).**

Parte I – Apresentação do TOP 10

Figura 3 - Top 10

1> Pretende-se apresentar a lista do TOP 10, ao clicar em "Ver TOP 10", como apresentado na figura 3.

b. Para isso, declare o array **`topGamers`**, no *scope* global, e inicialize-o com dois objetos com duas propriedades, como se apresenta de seguida.

```
let topGamers = [
                    { nickname: 'Ze', points: 331 },
                    { nickname: 'Maria', points: 321 }
                ]
```

c. Implemente a função `getTop10` que irá mostrados os dados do *array* na janela TOP 10. Para isso:

- Declare a variável `infoTop` que deve aceder ao elemento da página cujo id é `infoTop`, necessária para apresentar a lista dos jogadores.
- Implemente um **ciclo** que percorra o array **`topGamers`** e construa uma *string* com os valores dos elementos existentes no array, no seguinte formato ***nickName – points***.
- Declare uma variável **gamers** destinada a armazenar os valores (nickname; points) dos diversos jogadores.
- Com recurso à variável `infoTop` e à propriedade `innerHTML`, escreva a *string* construída no painel do top10, como se apresenta na Figura 4.
- Adicione um *event listener* ao **`btTop`** (não necessita de definir btTop pois já existe essa constante definida em *modal.js*), de forma a invocar a função `getTop10` quando há um click no botão TOP10.

Figura 4- Janela Modal - Top 10 (inicial)

- Confirme no **browser** o comportamento.

d. Como pode verificar, o aspeto da janela TOP 10 deve ser substancialmente melhorado. Para tal, implemente os seguintes passos:

- Comente o código implementado na alínea anterior, mais propriamente, a criação da *string* para apresentação dos jogadores no top10.
- De forma a impedir a visualização repetida do painel de resultados, sempre que a função é invocada deve limpar o conteúdo de infoTop, recorrendo para tal a:

```
infoTop.textContent = "";
```

- Crie elementos HTML, manipulando o DOM, de forma a obter a estrutura apresentada abaixo:

```
▼<div class="info" id="infoTop">
  ▼<div>
     <p>Nick Name</p>
     <p>Pontuação</p>
   </div>
  ▼<div>
     <p>Ze</p>
     <p>331</p>
   </div>
  ▼<div>
     <p>Maria</p>
     <p>321</p>
   </div>
 </div>
```

  - Como pode verificar, cada jogador (objeto) corresponde a um **bloco div com dois parágrafos**, um para o nickname e outro para a pontuação.
  - Substitua o código anterior de forma a criar estes elementos dinamicamente com recurso aos métodos e propriedades para manipulação do DOM como efetuado para criação do tabuleiro de jogo:
    - `createElement()`
    - `appendChild()` `elemento.cloneNode(true)`
    - `textContent`
    - `firstChild / lastChild`
  - A listagem do top10 deve ser implementada de acordo com o representado na figura 1.

- Confirme no browser a visualização do TOP 10 que já deverá ter o aspeto desejado.

2> Pretende-se apresentar a "Pontuação TOP" quando se inicia o jogo, como apresentado na figura. Para isso:

a. Implemente a função `getTopPoints` que deve apenas obter os pontos do primeiro objeto existente no array e escrever os pontos elemento cujo ID é `pointsTop`.

b. Invoque a função anterior em `startGame`.

c. Confirme o resultado no browser.

Figura 5 - Apresentação da Pontuação TOP

Parte II – Gestão do TOP 10

3> Nesta fase, pretende-se verificar se o utilizador entra no TOP 10. Em caso afirmativo, é apresentado um campo para introdução do *nick name*, como apresentado na Figura 6.

Para isso, implemente os seguintes passos:

a. Implemente a função `getLastPoints,` no *scope global,* que deverá retornar apenas a pontuação do último jogador existente no *array* dos topGamers.

b. Implemente a função `checkTop10` que verifica se o jogador entra ou não no top 10 e, se entrar, deve ser apresentada a caixa para especificar o seu **`nickname`**. Tenha em consideração:

Figura 6 - Janela Modal - Jogo Terminado

- Para um utilizador poder entrar no TOP10, deve garantir uma das condições: se ainda não existirem 10 jogadores na lista ou então, se existir, a pontuação é superior à última.
- Recorra à função `getLastPoints` implementada anteriormente para obter o número de pontos obtidos pelo último utilizador.
- A função `checkTop10` deve ser invocada na função `stopGame`

c. Para mostrar a caixa de diálogo do nickname, deve:

- Declarar a variável nick que permite aceder ao elemento da página cujo id é nickname. A visualização deve ser baseada na propriedade display = 'block'.
- No elemento com id **messageGameOver**, por recurso à propriedade innerHTML, acrescentar ao conteúdo já existente nesse elemento, o texto "
Parabéns! Entrou no TOP 10!".

d. Confirme o comportamento no browser.

4> Para gravar o nickname e os dados do utilizador, implemente os passos seguintes.

a. Implemente a função `saveTop10`:

- Criar um objeto com o nickname e pontuação. Para obter o *nickname*, deve aceder ao valor do elemento com id `inputNick.`
- Verificar se o jogador já se encontra na lista do *topGamers:*
  - Caso já exista, atualize a sua pontuação se a pontuação obtida no jogo for superior à existente no top.
  - Se não existir, adicione o elemento no fim do array..
- Ordenar o array através da pontuação dos elementos, da seguinte forma:

```
topGamers.sort(function (a, b) { return b.points - a.points });
```

- Se existirem mais de 10 elementos no array, eliminar o último.

b. Implemente o *eventListener* que invoque a função anterior, quando existir um click no botão cujo id é `okTop`. Para além de invocar a função `saveTop10`, deve fechar a janela modal, recorrendo ao código **modalGameOver.close() assim como** invocar a função `reset`.

c. Confirme o comportamento no browser e confirme que não existe erros na consola.

Parte III – localStorage

Nesta secção, pretende-se que os dados do Top10 persistam, ainda que a janela do browser seja fechada. A propriedade ***localStorage*** do JavaScript permitir implementar esse comportamento, no qual os métodos `setItem` e `getItem` permitem armazenar e aceder a dados locais, respetivamente.

O exemplo abaixo apresenta uma forma de armazenar e aceder a dados, simples ou objetos.

```
localStorage.setItem('meuDado', '1234');
localStorage.getItem('meuDado')

localStorage.setItem('meuObjecto',
JSON.stringify({prop:'1',prop:'2'}));
meuObjecto = JSON.parse(meuObjecto)
```

Armazenar o Top10 localmente é algo que não faz sentido numa situação real (p.e. num jogo online, com vários jogadores), no entanto, é usado neste contexto para fins educacionais, de forma que o aluno possa exercitar o uso da propriedade *localStorage*.

5> Para persistir os dados no jogo, implemente os seguintes passos:

a. Na função `saveTop10`, armazene os valores do *nickname* e a lista do top10, recorrendo ao método `setItem`.

b. Implemente a função `getLocalStorage` que deverá obter os dados armazenados com o *localStorage*, nomeadamente o *nickname* e a lista do TOP10.

c. Invoque a função anterior no *scope global*.

d. Verifique no browser o comportamento o jogo, **que deverá estar concluído**, de acordo com o pretendido nas aulas LS.

Parte IV – HTML Event Handlers

Na implementação do jogo em JavaScript, foram utilizados vários *event listeners* para interagir com os elementos do DOM. Para além deste método, que torna o código mais organizado e flexível, e considerado

uma boa prática, também é possível manipular eventos utilizando *HTML event handlers*, embora essa abordagem possa dificultar a manutenção de código.

6> Para explorar essa alternativa, substitua os *event listeners* existentes no jogo, por *HTML event handlers.*

a. Alteração do *DOM event listener por* um *HTML event handler* no elemento **btLevel**:

- Comente a linha de código `btLevel.addEventListener("change", reset);` que adiciona o *event listener;*
- Adicione a função `resetGame` ao seu código JavaScript:

```
const resetGame = () => reset();
```

- No HTML, adicione o evento **onchange** ao elemento **btLevel** para invocar a função `resetGame`.

  Nota: A invocação reset não pode ser chamada diretamente no HTML, pois reset é uma palavra reservada, por isso, é criada a função resetGame como intermediária.

b. Alteração do *DOM event listener* para um *HTML event handler* no elemento **btPlay**:

- Comente o código relativo ao ***addEventListener*** correspondente;
- No HTLM, adicione o evento **onclick** ao elemento **btPlay** e adapte o código JS existente no *event listener* diretamente no atributo;

c. Alteração dos *event listeners* das cartas do tabuleiro:

- Comente todos os *event listeners* existentes

```
// card.addEventListener("click...
// card1.addEventListener("click...
// card2.addEventListener("click...
```

- Altere a função `createPanelGame` para adicionar o evento **onclick** às cartas criadas, utilizando o método **setAttribute("onclick", "flipCard(this)");**
- Por fim, ajuste a função `flipCard` para garantir que o jogo continue funcional. Certifique-se de que a função *flipCard* receba o elemento clicado como parâmetro (this) e processe o clique corretamente. Além disso, garanta na função que a rotação da carta só existe após início do jogo e caso a mesma ainda não tenha a class **flipped** aplicada.

d. Verifique no browser o comportamento o jogo, **que deverá estar concluído**, sem recorrer a Event Listerners.