

> Ficha Prática Nº2 (JavaScript – Varáveis, Event Listeners, Funções)

Notas:

- Os alunos **não devem alterar** o documento HTML nem os ficheiros de estilos existentes, de forma a seguirem o propósito da ficha.
- **Não devem remover** a instrução `'use strict'` que se encontra no topo do ficheiro `index.js` de forma que seja usada na implementação, uma variante mais restrita do *JavaScript*.
- Pretende-se, nas próximas aulas, implementar, em JavaScript, o tradicional jogo de memória que terá o aspeto apresentado das figuras seguintes. Recomenda-se, para cada ficha, ler o apoio à sua resolução (apresentadas na seção **Apoio**) antes de iniciar resolução da ficha. O HTML, bem como o CSS necessário à resolução, já inclui todos os elementos necessários.



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

> Apoio à resolução da ficha:

- a. Concentre a declaração das variáveis globais no topo do **index.js**, depois da instrução **'use strict';** por forma a facilitar a sua localização.
- b. Variáveis **que não serão alteradas**, devem ser declaradas com **const**, caso contrário, declare com **let**. Recomenda-se a não utilização da declaração de variáveis com recurso ao **var**.
- c. Existem várias formas para aceder a um elemento DOM, seja para alterar o seu estado ou para adicionar um *event listener*. Exemplos:

<code>document.querySelector('#elemento')</code>	> Permite obter o elemento cujo id é elemento
<code>document.querySelector('.elemento')</code>	> Permite obter o 1º elemento com a classe elemento
<code>document.getElementById('elemento')</code>	> Permite obter o elemento cujo id é elemento
<code>document.querySelectorAll('.elemento')</code>	> Permite obter todos os elementos com a classe elemento

- a. A **alteração dos estilos** de um elemento pode ser efetuada com recurso à propriedade *style*, da seguinte forma:

```
elemento.style.display = 'none';
```

- b. A **alteração do texto** de um elemento pode ser efetuada com recurso à propriedade:

```
elemento.textContent = 'JavaScript';
```

- c. A **propriedade classList** permite aceder às classes associadas a um elemento. Com esta propriedade, e com recurso aos métodos `add()`, `remove()` e `toggle()`, é possível adicionar, remover ou alternar entre duas classes, respetivamente.

```
elemento.classList.add("estilo");
```

- d. A sintaxe genérica para definir um *event listener* é a seguinte:

```
elemento.addEventListener(e, function, useCapture)
```

- > **elemento** – Elemento que se está a associar o evento;
- > **e** – Evento a capturar (ex: click)
- > **function** – Função a ser executada
- > **useCapture** – Parâmetro opcional que indica se deve haver encadeamento de eventos

O método *addEventListener* permite anexar um *event handler* a um determinado elemento.

e. Existem várias formas de aplicar o método **addEventListener()**, como apresentado nos exemplos abaixo:

- **Declarando uma função externa** e especificando como argumento o nome dessa função. Esta função pode ser invocada por outros *event listeners* ou outras funções.

Nota Importante: o nome da função externa é especificado sem parêntesis.

```
const elemento = document.querySelector('h1');
elemento.addEventListener('click', exemplo);

function exemplo() {
    console.log('O elemento h1 foi clicado!');
}

// ou então
function exemplo(event) {
    console.log(`O elemento ${event.target.tagName} foi clicado!`);
}
```

- Recorrendo, **a uma função sem nome** (*anonymous function*) na qual se efetua uma chamada à função externa.

```
const elemento = document.querySelector('h1');
function exemplo(msg) {
    console.log(msg);
}
elemento.addEventListener('click', function () {
    exemplo('Elemento Clicado! Funcao com parametros!');
});

// outra forma...
elemento.addEventListener('click', () =>
    exemplo('Elemento Clicado! Funcao com parametros!'));
```

- Recorrendo, **a uma função sem nome** (*anonymous function*) na qual implementa o processamento pretendido.

```
const elemento = document.querySelector('h1');

elemento.addEventListener('click', function () {
    console.log('O elemento foi clicado!');
});

// ou então com arrow function
elemento.addEventListener('click', () => {
    console.log('O elemento foi clicado!')
});
```

f. A sintaxe genérica para definir uma estrutura de controlo de repetição com **for..of** é a seguinte:

```
for (variavel of iteravel) {
    //... código ser executado
}
```

g. A sintaxe genérica para definir uma estrutura de controlo de repetição com **forEach** é a seguinte:

```
elementos.forEach(function(elemento, index, arr)) {
    //...
});
```

- > **function** – função a ser executada por cada elemento
- > **index** – opcional, índice do elemento corrente
- > **arr** – opcional, array do elemento corrente

> Preparação do ambiente – Efetue os seguintes passos:

- a. **Descompacte** o ficheiro **ficha2.zip**.
- b. Inicie o *Visual Studio Code*, abra a **pasta no workspace** e visualize a página **index.html** no browser, na qual terá o aspeto da figura 2.
- c. Como pode verificar, a figura não apresenta o aspeto necessário para iniciar o jogo, e será esse o propósito da ficha, escondendo/mostrando e ativando/ desativando elementos do jogo, recorrendo a *event listeners*.
- d. Analise o código HTML, bem como as classes existentes no ficheiro **estilos.css**, para resolução do jogo de memória pretendido.
- e. Durante a realização das várias fichas, aceda às *Developer Tools* do browser (ex. clicando F12 no Chrome) e selecione a tab “Console” onde poderá ver os erros de *Javascript* da página ou simplesmente utilizar essa área como um editor de *Javascript*.



Figura 2 - Jogo (início)

Parte I – Implementação função Reset

- 1> Nesta fase, pretende-se criar uma função que vai especificar o aspeto inicial do jogo. A figura 3 apresenta o aspeto pretendido. Para facilitar a implementação, siga os passos e consulte a secção de apoio anteriormente apresentada.

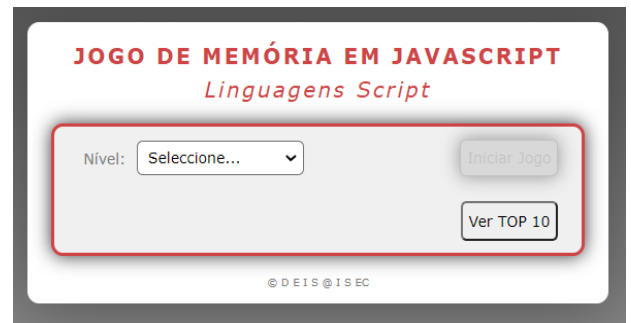


Figura 3 – Início do Jogo

- a. Especifique, no ficheiro **index.js**, as seguintes **variáveis** contantes, logo após a declaração **'use strict'** existente no ficheiro:

- **panelControl** que permite aceder ao elemento **#panel-control**
- **panelGame** que permite aceder ao elemento **#game**
- **btLevel** que permite aceder ao elemento **#btLevel**
- **btPlay** que permite aceder ao elemento **#btPlay**
- **message** que permite aceder ao elemento **#message**

- b. Implemente a função **reset()** no ficheiro **index.js** de forma a completar os seguintes passos:

1. Deverá esconder o *panelGame*, especificando propriedade **display** com **none**;
2. Invoque a função reset, através da respetiva chamada **reset()** e confirme o resultado no *browser*, certificando-se que não existe nenhum erro de código JavaScript.
3. Elimine o texto existente no elemento **message**, especificando a propriedade **textContent=""**, e **coloque o elemento visível**, removendo a class **hide** que se encontra aplicada. Para isso, recorra à propriedade **classList** e método **remove**.

```
message.classList.remove('hide');
```

4. Implemente o código de forma que o elemento **#btPlay** fique desativado. A propriedade que permite esse comportamento é **disabled = true**

- c. Confirme, no *browser*, se o resultado tem o aspeto da figura 3.

Parte II – Implementação de Event Listeners

- 2> Nesta secção, pretende-se especificar algum comportamento, quando se seleciona/clica em alguns elementos. Assim, sempre que se selecionar um nível (**btLevel**), o botão de iniciar jogo (**btPlay**) fica ativo, caso contrário, fica desativo, como mostram as figuras seguintes.

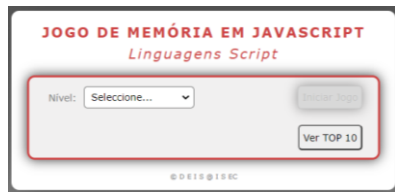


Figura 4 - Opção '0' selecionada



Figura 5 - Opção '1' selecionada

- a. Para implementar este comportamento, altere a declaração implementada na função **Reset** no qual o botão “Iniciar Jogo” (**#btPlay**) fica **sempre desativo** se a opção selecionada for “Selecione...”, opção cujo valor é **'0'** (podem confirmar no html), caso contrário, se for selecionado um dos níveis de jogo, o elemento **#btPlay** **deve ficar ativo**.

Para obter o valor da opção selecionada no elemento *select*, deve recorrer à propriedade **value**.

- b. Também o **panelGame** deverá ficar visível quando se seleciona um nível, aplicando para isso o estilo **grid** à propriedade **display**, caso contrário, deverá manter-se escondido.
- c. Implemente um *Event Listener* de forma a que, sempre que houver uma alteração à opção selecionada (evento **change**) no elemento **btLevel**, seja executada esta função **reset**.
- d. Confirme, no *browser*, se o resultado tem o comportamento pretendido.

- 3> Tendo em consideração tudo o que implementou nas alíneas anteriores, implemente o código necessário de forma que, quando se clica no botão “Iniciar Jogo”, o seu texto mude para “Terminar Jogo” e apresente os vários elementos com os dados do jogo, como se apresenta na figura seguinte.



Figura 6 - Jogo Iniciado

Para obter o comportamento e aspeto da figura 6, complete os seguintes passos:

- a. Implemente uma função de nome **startGame** que deve realizar os seguintes passos:
 1. Inativar o elemento que permite a seleção do nível (**btLevel**)
 2. Especificar o texto “Terminar Jogo” ao botão **btPlay**
 3. Adicionar a class **hide** ao elemento **message**
 4. Adicionar a class **gameStarted** a todos elementos especificados com a classe **.list-item**.

Para isso:

- Crie uma variável que obtenha todos os elementos especificados com a classe **.list-item** existentes no **painelControl**. Para isso, deve usar o **querySelectorAll**.
- Com recurso ao **for... of** ou com o **forEach**, percorra todos os elementos obtidos na alínea anterior de forma a adicionar a class **gameStarted**. Deve usar a propriedade **classList** e o método **add**.

```
<div class="form-metadata">
  <!--class="hide"-->
  <p id="message" role="alert">Clique em Iniciar o Jogo!</p>

  <dl class="list-item left">
    <dt>Tempo de Jogo:</dt>
    <dd id="gameTime">0</dd>
  </dl>

  <dl class="list-item right">
    <dt>Pontuação TOP:</dt>
    <dd id="pointsTop">0</dd>
  </dl>

  <dl class="list-item left">
    <dt>Pontuação:</dt>
    <dd id="points">0</dd>
  </dl>
  <div id="top10" class="right">
    <button id="btTop">Ver TOP 10</button>
  </div>
</div>
```

- b. Implemente uma função de nome **stopGame** que deve realizar os seguintes passos:
 1. Especificar o texto “Iniciar Jogo” ao botão **btPlay**
 2. Ativar o elemento que permite seleção do nível (**btLevel**)
 3. Invocar a função **Reset**
- c. Implemente um *Event Listener* para o elemento **btPlay** de forma a que quando se clica nele:
 - Se o texto for “Terminar Jogo”, deve invocar a função **stopGame**
 - Caso contrário deve invocar a função **startGame**
- d. Adicione à função **reset**, código para que remova a class **‘gameStarted’** a todos elementos especificados com a classe **.list-item**, de forma a que os elementos como o tempo e pontuação de jogo voltem a ficar escondidos, sempre que se cancela ou termina um jogo. Assim, para todos os elementos **list-item**, remova a class **‘gameStarted’** com recurso à propriedade **classList** e o método **remove**.
- e. Confirme, no *browser*, se o resultado final tem o comportamento pretendido. Confirme ainda que na consola não existem erros de JavaScript.

- 4> Para concluir, especifique o código necessário para que, quando se clicar no panel de jogo **panelGame** não se encontre a mensagem “Clique em Iniciar Jogo!”, presente, caso contrário, remova-a. A figura 7 apresenta o aspeto desejado. Para implementar o pretendido, apenas necessita de adicionar um *event listener* ao elemento **panelGame** e verificar se existe texto ou não.



Figura 7 – Mensagem “Clicar em Iniciar jogo”