> *Ficha Prática Nº5 (Jogo de Memória – Pontuação e Nº de Cartas)*

Esta ficha tem como objetivo implementar as funções necessárias para **calcular e atualizar a pontuação obtida em cada jogada** (encontrar um par/ não encontrar um par) e apresentar a pontuação final numa nova janela quando o jogo termina. Pretende-se ainda, implementar as funções necessárias para especificar **o nº de cartas do tabuleiro de acordo com o nível de jogo.**

A figura 1 apresenta várias imagens com o resultado da ficha.

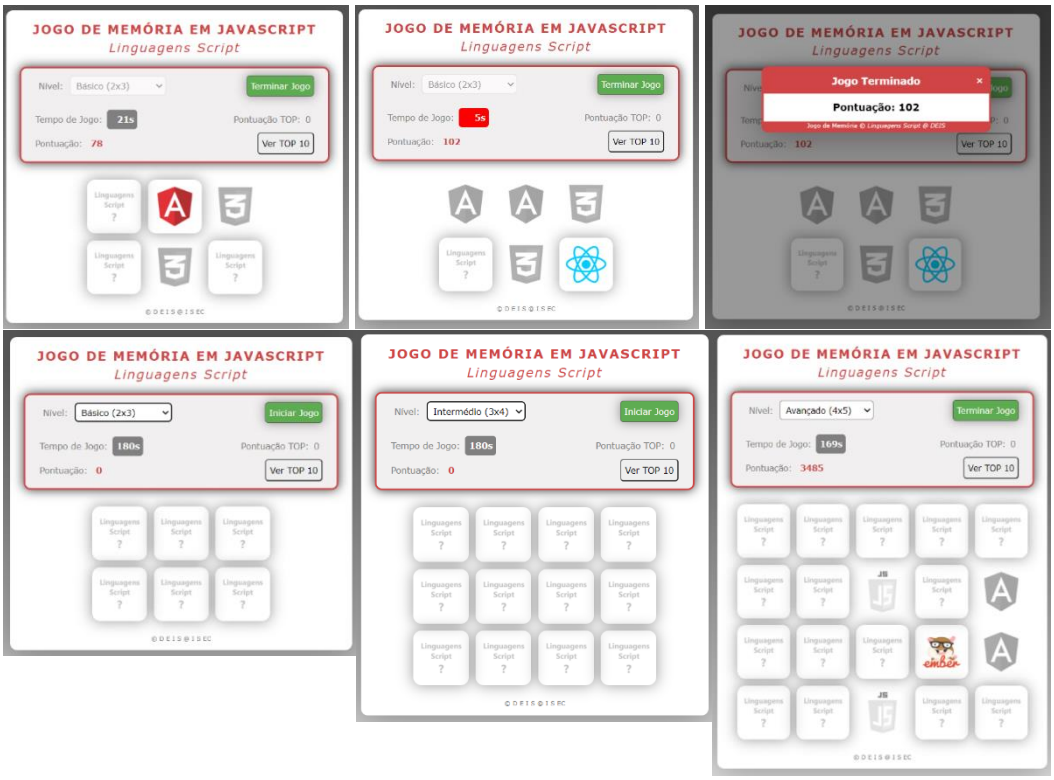


Figura 1 – Ficha 5 – Aspeto final

> *Preparação do ambiente*

a. Descompacte o ficheiro **fichaPratica_05.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.

b. Inicie o *Visual Studio Code* e abra **a pasta no *workspace*.** Visualize a página **index.html** no browser.

c. Módulos a implementar:

- **Cálculo da pontuação (Parte I)**, que se altera em cada jogada, aumentando quando é encontrado um par de cartas ou diminuindo quando jogador falha o par.
- **Apresentar modal com pontuação (Parte I > 2)**, quando o jogo termina ou é interrompido.
- **Gerar tabuleiro (Parte II)** de acordo com o nível selecionado.

Parte I – Calcular Pontuação do Jogo

A lógica para a pontuação poderá ser (ou outra se assim o desejar):

- **Quando um par é encontrado:** A pontuação dessa jogada é obtida pelo **produto entre o tempo de jogo e o número de pares que falta encontrar para concluir o jogo** (por exemplo: no nível básico, devem ser encontrados 3 pares e descobriu o primeiro par, logo será o tempo * 2). Isto irá permitir obter tanto mais pontuação, quanto maior for o número de peças no tabuleiro (nível) e quanto mais rápida for a identificação dos pares. Na última jogada, em que todos os pares foram identificados, garanta que adiciona o tempo no qual o jogo foi concluído, à pontuação final.
- **Quando a escolha de par é falhada:** Deve ser efetuada a **subtração no valor de 5 pontos**, à pontuação total existente. Claro que, caso a pontuação total seja inferior a 5 pontos, a pontuação passa a ser 0.

1> Para calcular a pontuação, implemente os seguintes passos:

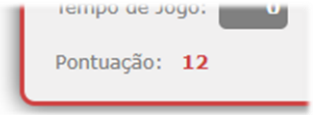
a. Declare a variável `labelPoints` que deve aceder ao elemento da página cujo id é `points`, onde será apresentada ao jogador, a pontuação.

b. Declare a variável **`totalPoints`**, no *scope global,* que irá armazenar a pontuação.

c. Inicialize esta variável a 0, sempre que inicia um jogo, portanto na função **`startGame. Garanta que o valor da pontuação a 0 é atualizado em labelPoints.`**

d. Implemente a função **`updatePoints`**, que deverá atualizar a pontuação do jogo:

- A função deverá receber por parâmetro a operação a efetuar (se for '+', é para somar pontuação de acordo com as regras especificadas no início desta secção; se for '–' é para subtrair).
- Deverá atualizar a pontuação em `labelPoints` com a propriedade `textContent`



e. Invoque a função **`updatePoints`** na função **`checkPair`**

f. Confirme no browser o comportamento do jogo.

2> Quando o jogo termina, a janela de fim de jogo deve surgir. Para isso, implemente os seguintes passos na função ***stopGame***:

- Apresente a pontuação obtida no elemento com id ***messageGameOver*** com recurso à propriedade `textContent`.

- Esconda a área para especificar o nome, especificando a propriedade display:none, no elemento cujo id é **nickname**
- Coloque em comentário a invocação à função o **reset**, de forma ver, em imagem de fundo, o estado do tabuleiro. Na realidade, a função **reset** será invocada automaticamente quando a janela modal for fechada (pode analisar este código, já implementado, no ficheiro *modal.js*)

Parte II – Criação de Tabuleiro

A criação do tabuleiro de jogo de acordo com nível selecionado (maior nível -> maior número de cartas), será efetuada com recurso ao JavasSript. Se o nível selecionado é **básico**, devem ser apresentadas 6 cartas (3 pares), se **intermédio**, o tabuleiro deve ser composto por 12 cartas (6 pares) e, por fim, se **avançado**, o número de cartas deve ser 20 (10 pares).

Podem ser usadas várias técnicas para criação do tabuleiro de jogo, no entanto, pretende-se implementar este módulo recorrendo aos métodos e/ou propriedades de manipulação do DOM.

1> Especifique a função `createPanelGame` e invoque-a na função `reset`. Esta função irá criar o tabuleiro, de acordo com o nível selecionado. Para isso, implemente os seguintes passos.

a. Elimine todas as peças existentes no tabuleiro **panelGame**. Este comportamento pode ser implementado com o código:

```
panelGame.innerHTML = '';
```

b. Implemente o código necessário para gerar o código HTML de **uma** carta. Para isso recorra a vários métodos e/ou propriedades de manipulação do DOM, entre eles:

- `document.createElement(elemento)` - cria um *elemento*;
- `elemento.appendChild(elFilho)` - anexa o *elFilho* como último filho do *elemento*.
- `elemento.setAttribute(nome, valor);` - adiciona um novo atributo ou modifica o valor de atributo existente.

Por exemplo, o código abaixo, à esquerda, apresenta o *JavaScript* para gerar o HTML apresentado

```javascript
let newDiv = document.createElement('div');
newDiv.setAttribute('class', 'card');
let imgBack = document.createElement('img');
imgBack.setAttribute('src', 'images/ls.png');
newDiv.appendChild(imgBack);
panelGame.appendChild(newDiv);
```

```html
<div class='card'>
  <img src='images/ls.png'>
</div>
```

Assim, copie o código JS anterior e efetue as alterações necessárias, para completar a geração de uma carta do tabuleiro, que deve ter o seguinte código HTML:

```html
<div class="card">
      <img src="images/ls.png" class="card-back" >
       <img class="card-front">
</div>
```

c. Duplique o **div** anteriormente criado, tantas vezes quantas necessárias, de acordo com o nível de jogo selecionado, **Básico** - 6 cartas | **Intermédio** - 12 cartas | **Avançado** – 20 cartas.
Esta duplicação de elementos pode ser efetuada com recurso ao `elemento.cloneNode(true)` que retorna a cópia de um elemento, incluindo todos os seus filhos, atributos e valores.

d. No fim da função, atualize a variável `cards` de acordo com o novo `panelGame`. Assim, deve especificar o código em que a propriedade `childNodes` permite obter todos os nodos de um elemento.

```javascript
cards = panelGame.childNodes;
```

*Nota: Certifique-se que a variável **cards** não é constante.*

e. Adapte, na função `startGame`, o número de elementos no *array* ***newCardLogos*** para qualquer nível de jogo. Assim, substitua o limite de 3 por `cards.length/2`.

f. **Confirme no browser o jogo com os vários níveis.**

g. De forma a que as cartas sejam distribuídas de forma diferente no tabuleiro (relativamente de nº de cartas por linha/coluna), dependendo do nível de jogo, adicione, na função **createPanelGame**, as seguintes classes (já existentes no css) ao **panelGame**:

- Se nível intermédio, adicionar classe `intermedio` (especifica *grid* 3x4)
- Se nível avançado, adicionar classe `avancado` (especifica *grid* 4x5)

O nível básico, sendo o nível por omissão, já tem especificada uma grid de *2x3*

h. Para que qualquer das classes anteriores, seja removida do panelGame, sempre que se gera novo tabuleiro, aplique seguinte código (no início da função *createPanelGame*):

```
panelGame.className = '';
```

i. Confirme no browser o jogo, o qual deverá ter o aspeto das seguintes figuras.

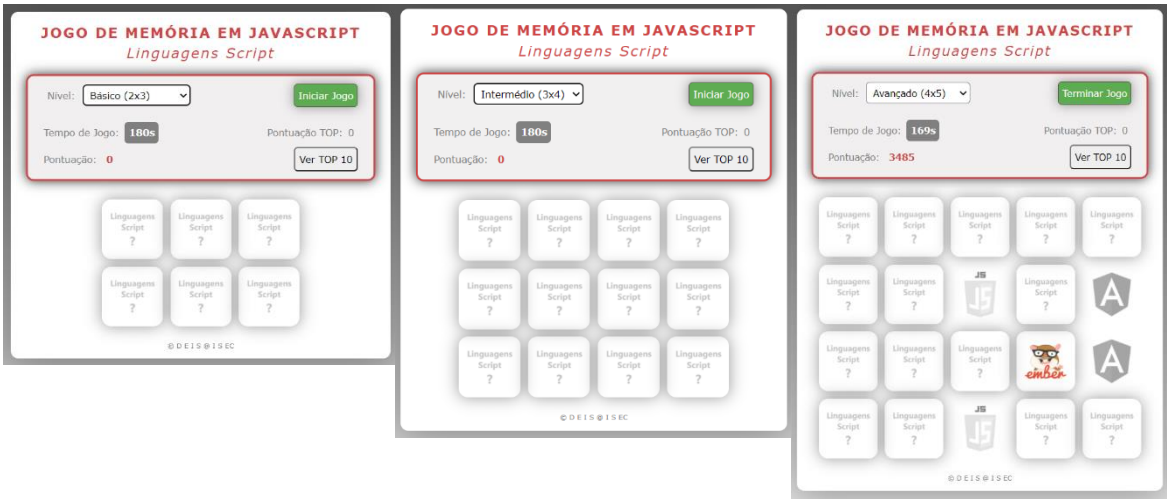


Figura 2 - Ficha 5