> Ficha Prática Nº5 (Jogo de Memória JavaScript – Continuação)

- Esta ficha, tem como objetivo implementar as funções necessárias para especificar o **tempo de jogo**, **calcular e atualizar a pontuação nas jogadas** (caso encontre par ou caso falhe) e apresentar a pontuação quando o jogo termina. Por fim, pretende-se **criar o painel de jogo de forma dinâmica**, de forma a que se adapte a qualquer nível.
- O resultado final da ficha apresenta-se nas figuras seguintes.

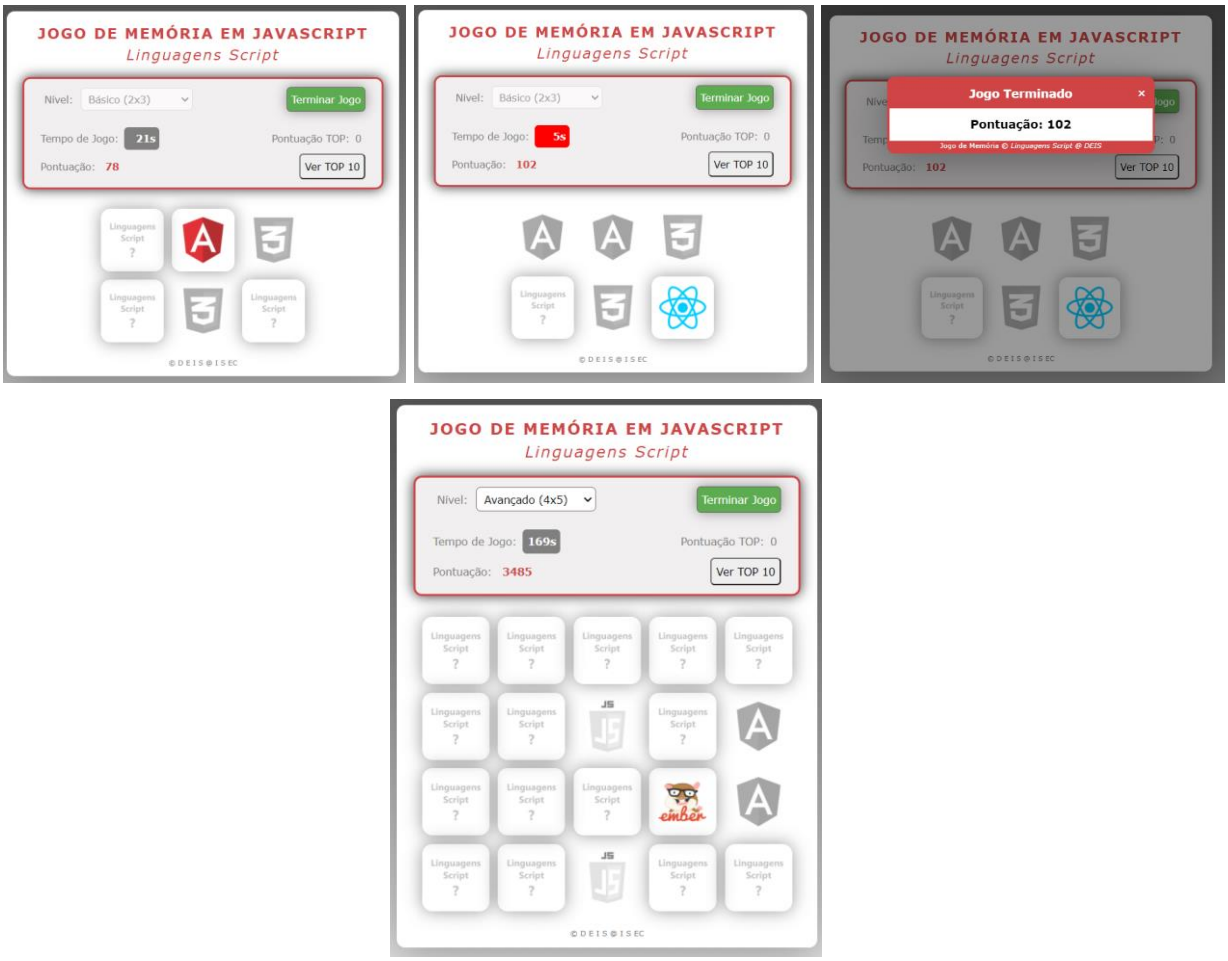


Figura 1 – Ficha 5 – Aspeto final

> Preparação do ambiente

a. Efetue o download e descompacte o ficheiro **ficha5.zip** disponível no *inforestudante*.

NOTA: Os alunos que concluíram a resolução da ficha 4, podem usar essa versão.

Por curiosidade, podem analisar o JavaScript fornecido nesta ficha.

b. Inicie o *Visual Studio Code*, abra **a pasta no *workspace*** e visualize a página **índex.html** no browser.

Parte I – Tempo de Jogo

Pretende-se especificar o código necessário para existir um tempo de jogo limitado, no qual o tempo inicia com **180s** (segundos) e decrementa até chegar aos 0s. Nos últimos 10 segundos, a cor de fundo onde o tempo é apresentado deverá ser destacado. As figuras seguintes apresentam o comportamento pretendido.

Figura 2 - Tempo de Jogo

1> Para especificar o tempo de jogo será necessário recorrer ao método `setInterval.` É uma função assíncrona, o que quer dizer que a função do tempo não irá parar a execução de outras funções, e permite **executar código ou invocar uma função repetidamente**, com um t**empo de espera fixo entre cada execução. No caso do jogo, pretende-se alterar o apresentar um tempo de jogo diferente, a cada segundo.** O `setInterval` pode ser cancelado recorrendo ao `clearInterval()`.

a. Implemente os seguintes passos:

- Defina a constante `TIMEOUTGAME` com o valor 20 (considerando numa fase inicial que o jogo irá durar 20 segundos).
- Defina a varivel `labelGameTime` que deve aceder ao elemento da página cujo id é `gameTime.` Será necessário para atualizar o tempo de jogo.
- Defina a variável **`timer`**, que irá armazenar o tempo de jogo. Note que este timer, irá ser alterado durante o jogo.
- Defina a variável **`timerId`**, que irá armazenar o identificador para o tempo de jogo, quando se especificar a função `setInterval`
- Na funçao `startGame`:
  - Inicialize a variável **`timer`** com o valor da constante `TIMEOUTGAME.`
  - Especifique o `setInterval` da seguinte forma.

```
timerId = setInterval(updateGameTime, 1000)
```

  - Repare que a função `setInterval` irá executar uma função **`updateGameTime`**, repetidamente, a cada segundo.

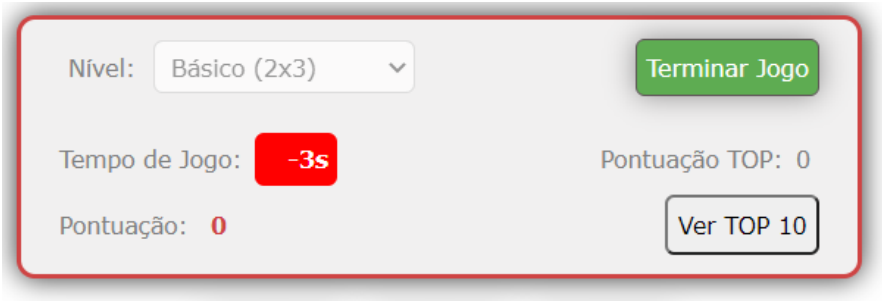
- Crie a função **updateGameTime:**
  - Esta deverá decrementar a variável **timer**
  - Deve atualizar o tempo em `labelGameTime` recorrendo à propriedade `textContent`

- Na função **stopGame**:
  - Adicione a seguinte linha de código, para que este temporizador seja cancelado quando o jogo termina:

```
clearInterval(timerId);
```

- **Verifique no browser o comportamento do jogo**.

b. Como pode verificar, depois de atingir o tempo de jogo 0, e o jogo ainda continuar, a função **updateGameTime** continua a decrementar, como se mostra na figura seguinte.



- Assim, de forma a que este comportamento não aconteça, é necessário especificar código para que quando o **tempo** chegar a 0, o jogo seja cancelado.
  - Na função **updateGameTime**, invoque a função **stopGame** quando o tempo for 0.

- **Verifique no browser o comportamento do jogo**, que deverá ter o desejado.

2> Altere a cor de fundo do elemento `labelGameTime` para vermelho, quando o timer for inferior a 10 segundos.

a. Implemente o código necessário na função `updateGameTime`., alterando a propriedade style, com a cor de background para vermelho.

b. Note que, quando faz um reset, a cor deverá voltar ao estado normal (na função `reset`), isto é, basta remover o atributo style ao elemento, por exemplo, com recurso ao método `removeAttribute`

```
labelGameTime.removeAttribute('style');
```

c. Além disso, adicione na função reset o tempo de jogo no elemento `labelGameTime`.

d. **Confirme no browser o comportamento do jogo, o qual já deverá ter o tempo a funcionar**

Parte II – Calcular Pontuação

Nesta fase, pretende-se especificar o código necessário para calcular a pontuação nas jogadas. A lógica para a pontuação deve ser a seguinte:

- **Quando um par é encontrado:** A pontuação dessa jogada é obtida pela **multiplicação do tempo de jogo pelo número de pares existentes no jogo** (por exemplo: no nível básico, devem ser encontrados 3 pares, logo será o tempo * 3). Isto irá permitir obter mais pontuação quanto maior for o nível e mais rápido foi encontrado um par.
- **Quando a escolha de par é falhada:** Deve ser efetuada uma **subtração no valor de 5 pontos**, à pontuação total existente. Claro que, caso a pontuação total for inferior a 5 pontos, a pontuação passa a 0.

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

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

b. Defina a variável **`totalPoints`**, que irá armazenar a pontuação.

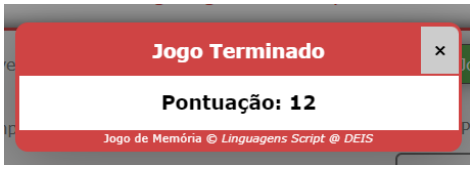
c. Inicialize a 0 a variável na função **`startGame`**

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

- A função deverá receber por parametro 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).
- Deve atualizar o tempo em `labelPoints` recorrendo à propriedade `textContent`
- Invoque corretamente esta função na função **`checkPair`**

e. Confirme no browser o comportamento do jogo.

4> Quando a janela modal de fim de jogo aparece, escreva a pontuação obtida de forma a ficar com o aspeto da figura seguinte.



- Assim, aceda ao elemento da página com id ***`messageGameOver`*** e com recurso à propriedade `textContent` apresente a pontuação.

Parte III – Criação de Painel Dinâmico

Pretende-se especificar o código necessário para que o painel de jogo seja criado de forma dinâmico, isto é, quando o nível for básico, devem ser usadas 6 cartas, se o nível for intermédio, o painel deve ser composto por 12 cartas e, por fim, se for avançado, o numero de cartas deve ser 20.

5> Para implementar o painel dinâmico, será necessário recorrer a um conjunto de métodos e propriedades para manipulação do DOM, nomeadamente:

- `createElement()` - método que permite criar um elemento.
- `appendChild()` - método que permite anexar um elemento (nó) como o último filho de um elemento.
- `elemento.cloneNode(true)` – método que permite criar uma cópia de um elemento e de todos os filhos, os atributos e valores. Retorna o elemento clonado.
- `childNodes` – propriedade que permite obter todos os nodos de um elemento.

O exemplo abaixo, cria uma **div** com class='card', com um **img** dentro dessa **div**.

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

Assim, o HTML gerado para este bloco de código é:

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

6> Para implementar o painel dinâmico, implemente os seguintes passos:

a. Crie a função `createPanelGame()` que deverá ser invocada na função `reset`.

b. O primeiro passo para criar o novo painel de jogo, é eliminar todas as cartas existentes no painel. Assim, especifique o seguinte código.

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

c. Implemente o código necessário para criar uma carta, de forma a que seja gerar o seguinte HTML.

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

d. Com recurso ao método `cloneNode(true)`, duplique o div anteriormente criado o número de vezes necessário, tendo em conta que:

- Básico - 6 cartas
- Intermédio - 12 cartas
- Avançado – 20 cartas.

e. No fim da função, atualize a variável `cards` de acordo com o novo `panelGame`. Assim, deve implementar o código abaixo, e certifique-se que a variável **cards** não é constante, se for, altere de forma a poder ser alterada.

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

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

g. De forma a que a grid fique com melhor aspecto dependendo do nível de jogo adicione no **createPanel** as seguintes classes ao **painelGame**:

- Classe `basico` quando o nível é basico
- Classe `intermedio` quando o nível é intermédio
- Classe `avancado` quando o nível é avançado

h. Por forma a que seja feito o **reset** de todas as classes anteriormente aplicadas, no inicio da função aplique o seguinte código:

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

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

Figura 3 - Ficha 5