> Ficha Prática Nº5 (Jogo de Memória JavaScript – Tempo e Pontuação)

Esta ficha tem como objetivo implementar as funções necessárias para especificar o **tempo limite para cada jogo**, **calcular e atualizar a pontuação obtida em cada jogada** (caso encontre par ou caso falhe) e **apresentar a pontuação final quando o jogo termina**.

O resultado da ficha apresenta-se nas figuras seguintes.

Figura 1 – Ficha 5 – Aspeto final

A resolver:

- **Tempo de jogo**, que deverá iniciar com um valor (dependendo do nível) e ir reduzindo até chegar a 0, que termina o jogo;
- **Cálculo da pontuação**, 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**, quando o jogo termina ou é interrompido.

> Preparação do ambiente

a. Descompacte o ficheiro **ficha5.zip**.

Os alunos que concluíram a resolução da ficha anterior, devem continuar nesse trabalho. Por curiosidade, podem analisar o JavaScript fornecido nesta ficha. Os restantes alunos podem resolver esta ficha prática, tendo como base o código fornecido juntamente com esta 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 que o tempo de jogo apresentado ao jogador, seja alterado a cada segundo.** O `setInterval` pode ser cancelado recorrendo ao `clearInterval()`.

a. Para efetuar o comportamento desejado, implemente os seguintes passos:

- Declare a variável constante `TIMEOUTGAME`=20 (considerando que o jogo irá durar 20s).
- Declare a variável `labelGameTime` que deve aceder ao elemento da página cujo id é `gameTime`. Será necessário para poder atualizar o tempo de jogo na aplicação.
- Declare a variável **`timer`**, que irá armazenar o tempo de jogo e irá ser alterado durante o jogo.
- Declare a variável **`timerId`** para ser possível interromper a execução da função `setInterval` (a ver de seguida), quando necessário.
- Na função `startGame`:
  - Inicialize a variável **`timer`** com o valor da constante `TIMEOUTGAME`
  - Apresente o texto "**20s**" com recurso à propriedade textContent do `labelGameTime`
  - Especifique o `timerId` da seguinte forma

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

Note que a função `setInterval` irá executar a função `updateGameTime` (a implementar de seguida), a cada segundo (1000 = 1 segundo). Só irá parar, quando se cancelar este comportamento com a função `clearInterval`.

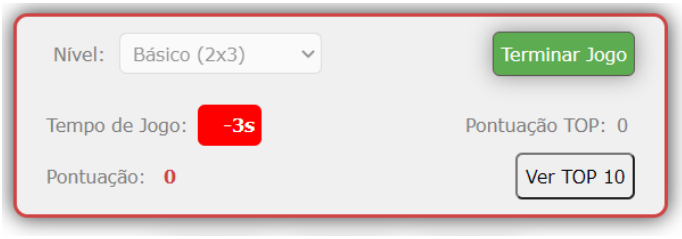
- Implemente a função `updateGameTime` que deverá:
  - Decrementar a variável `timer`
  - Atualizar o tempo em `labelGameTime` recorrendo à propriedade `textContent`

- Por fim, na função **stopGame**, cancele o temporizador, adicionando a seguinte linha de código, para que o tempo de jogo não seja novamente atualizado aquando término do jogo.

```
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 não tiver sido concluído, a função **updateGameTime** continua a decrementar, como se mostra na figura seguinte.



- Assim, para que isso não aconteça, é necessário invocar a função **stopGame** quando o tempo de jogo chegar a 0. Essa linha de código deverá ser então aficionada na função **updateGameTime**.
- **Verifique no browser o comportamento do jogo**, que deverá ser o desejado.

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

a. Altere a propriedade *style*, para vermelho no background, na função **updateGameTime**.

b. Na função `reset`, a cor deverá voltar ao estado normal, isto é, basta remover o atributo style ao elemento, por exemplo, com recurso ao método `removeAttribute`

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

c. Confirme no browser o comportamento do jogo.

3> Implemente as alterações necessárias para que o tempo de jogo, seja diferente, em cada nível. Para isso, implemente a função `getTimer()` que inicialize o temporizador (variável **timer**) de acordo com o valor das seguintes constantes:

- `TIMEOUTGAMEBASICO = 20;`
- `TIMEOUTGAMEINTERMEDIO = 60;`
- `TIMEOUTGAMEAVANCADO = 180;`

Confirme no browser o comportamento do jogo, o qual o temporizador já deverá estar a comportar-se corretamente e inicializado com valores diferentes, de acordo com o nível selecionado.

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 (ou outra se assim desejar):

- **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.

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

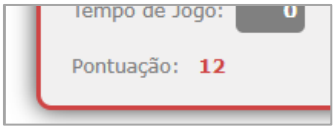
a. Declare a variável `labelPoints` que deve aceder ao elemento da página cujo id é `points`, onde será apresentada ao jogador, a pontuação.

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

c. Inicializa esta variável a 0, sempre que inicia um jogo, portanto na função **`startGame`**

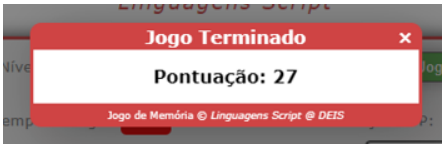
d. Implemente a função **`updatePoints`**, que deverá atualizar a pontuação do jogo:

- 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).
- Deve atualizar a pontuação em `labelPoints` recorrendo à propriedade `textContent`
- Invoque corretamente esta função na função **`checkPair`**



e. Confirme no browser o comportamento do jogo.

5> Quando a janela modal de fim de jogo aparece, escreva a pontuação obtida de forma a ficar com o aspeto da figura seguinte. Para isso, implemente os seguintes passos na função ***`stopGame`***:



- Apresentar a pontuação obtida no elemento com id ***`messageGameOver`*** e com recurso à propriedade `textContent`.
- Esconder a área para especificar o nome, especificando a propriedade display:none, ao elemento cujo id é **nickname**
- Comentar a invocação à função o **reset**, de forma ver em imagem de fundo, o estado do tabuleiro. O **reset** é posteriormente invocado quando a janela modal for fechada (poderá analisar este código no ficheiro *modal.js*)