# Cronómetro de conta atrás (exercicio guiado)

## Obxectivo

Practicar o manexo de temporizadores en JavaScript (`setInterval`, `clearInterval`) completando as partes marcadas con `TODO` nun cronómetro de conta atrás sinxelo.

## Como funciona a aplicación

Esta aplicación é un **cronómetro de conta atrás** que permite ao usuario:

1. **Configurar** cantos segundos quere contar (nun campo de entrada)
2. **Iniciar** a conta atrás premendo o botón "Iniciar"
3. **Ver** o tempo restante en formato mm:ss (minutos:segundos)
4. **Parar** o cronómetro en calquera momento co botón "Parar"

### Fluxo completo da aplicación

```
Usuario introduce segundos (ex.: 65)
   ↓
Preme "Iniciar"
   ↓
A función start() le os segundos do input
   ↓
Arranca un setInterval que cada 1 segundo:
   - Resta 1 segundo ao tempo restante
   - Actualiza a pantalla en formato "mm:ss"
   - Comproba se chegou a 0
   ↓
Cando chega a 0:
   - Detén o intervalo
   - Mostra "Tempo rematado"
```

### Exemplo práctico

- Usuario escribe **65** no input
- Preme "Iniciar"
- A pantalla mostra: **01:05** (1 minuto e 5 segundos)
- Cada segundo actualízase: 01:04, 01:03, 01:02...
- Ao chegar a **00:00**, mostra "Tempo rematado"

## Estrutura de ficheiros

```
ex01/
├── cronometro.html      # Interface (xa completa)
├── cronometro.css       # Estilos (xa completo)
```

```
├── cronometro.js        # JavaScript (TES QUE COMPLETAR OS TODOs)
└── enunciado.md         # Este ficheiro
```

## Que tes que completar (TODOs)

No ficheiro `cronometro.js` hai **4 funcións con TODOs** que tes que implementar:

### TODO 1: `formatMMSS(total)`

**Que fai**: Converte segundos totais en formato "mm:ss"

**Entrada**: un número (segundos totais)
**Saída**: un string con formato "mm:ss" (sempre 2 díxitos)

**Pasos a seguir**:

1. Asegurar que `total` sexa un enteiro non negativo (usa `Math.max(0, Math.floor(total))`)
2. Calcular minutos: divide entre 60 e redondea cara abaixo (`Math.floor(s/60)`)
3. Calcular segundos restantes: usa o operador módulo `%` (resto da división entre 60)
4. Converter ambos a string e engadir ceros á esquerda con `.padStart(2,'0')`
5. Devolver o resultado unindo ambos con `:` no medio

**Exemplos**:

- `formatMMSS(65)` → `"01:05"`
- `formatMMSS(5)` → `"00:05"`
- `formatMMSS(125)` → `"02:05"`

### TODO 2: `readDuration()`

**Que fai**: Le os segundos do input e devolve un valor válido

**Entrada**: ningunha (le de `segundosInput.value`)
**Saída**: un número enteiro positivo

**Pasos a seguir**:

1. Le o valor do input co `parseInt(segundosInput.value, 10)`
2. Comproba se é un número válido con `isNaN(val)`
3. Comproba se é negativo (`val < 0`)
4. Se é inválido ou negativo, devolve 10 por defecto
5. Se non, devolve o valor

**Exemplos**:

- Input ten "30" → devolve `30`
- Input ten "abc" → devolve `10` (non é número)
- Input ten "-5" → devolve `10` (é negativo)

### TODO 3: `start()`

**Que fai**: Inicia a conta atrás se non hai outra xa en marcha

**Pasos a seguir**:

1. Comproba se `intervalId` xa ten valor (se non é `null`), entón sae da función (`return`) para evitar intervalos duplicados
2. Obtén os segundos con `readDuration()` e gárdaos en `remaining`
3. Actualiza a pantalla chamando `updateTimeUI()`
4. Mostra unha mensaxe con `showMessage('Conta atrás en marcha')`
5. Arranca un `setInterval` que execute cada 1000ms (1 segundo):
    - Resta 1 a `remaining`
    - Actualiza a pantalla con `updateTimeUI()`
    - Comproba se `remaining <= 0`:
        - Se si: detén o intervalo con `clearInterval(intervalId)`
        - Pon `intervalId = null`
        - Mostra mensaxe "Tempo rematado"
6. Garda o ID do intervalo en `intervalId`

### TODO 4: `stop()`

**Que fai**: Detén a conta atrás e pon o tempo a 00:00

**Pasos a seguir**:

1. Comproba se hai un intervalo activo (`intervalId !== null`)
2. Se si, deteno con `clearInterval(intervalId)`
3. Pon `intervalId = null` para indicar que xa non hai intervalo
4. Pon `remaining = 0` para reiniciar o tempo
5. (Xa está feito) Actualiza a pantalla e mostra mensaxe

## Como probar a túa implementación

1. Abre `cronometro.html` no navegador
2. Introduce un valor no input (ex.: 10 segundos)
3. Preme "Iniciar" e observa:
    - O tempo debe mostrarse en formato mm:ss
    - Debe decrementar cada segundo
    - Ao chegar a 0, debe parar e mostrar "Tempo rematado"
4. Proba premer "Iniciar" varias veces seguidas: non debe crear intervalos duplicados
5. Proba premer "Parar": debe deter todo e poñer 00:00
6. Proba valores distintos: 5, 65, 125, 3600 (1 hora)

## Conceptos clave a practicar

`setInterval(función, milisegundos)`

Executa unha función de forma repetida cada X milisegundos e devolve un ID.

```
const id = setInterval(() => {
  console.log('Tic');
}, 1000); // cada 1 segundo
```

`clearInterval(id)`

Detén un intervalo usando o seu ID.

```
clearInterval(id); // Para o intervalo
```

### Operador módulo `%`

Dá o resto dunha división. Úsase para obter os segundos que sobran despois de sacar os minutos completos.

```
65 % 60  // → 5 (65 segundos = 1 minuto e sobran 5 segundos)
125 % 60 // → 5 (125 segundos = 2 minutos e sobran 5 segundos)
```

`padStart(lonxitude, caracter)`

Engade caracteres á esquerda dun string ata alcanzar unha lonxitude determinada.

```
"5".padStart(2, '0')  // → "05"
"15".padStart(2, '0') // → "15" (xa ten 2 caracteres)
```

## Criterios de avaliación

- `formatMMSS` formatea correctamente en mm:ss (20%)
- `readDuration` valida e devolve valores correctos (15%)
- `start` arranca o intervalo correctamente e evita duplicados (30%)
- `stop` limpa o intervalo e reinicia o estado (15%)
- O cronómetro funciona correctamente de principio a fin (20%)

## Entregables

Entrega o ficheiro `cronometro.js` coas funcións completadas segundo as especificacións anteriores.