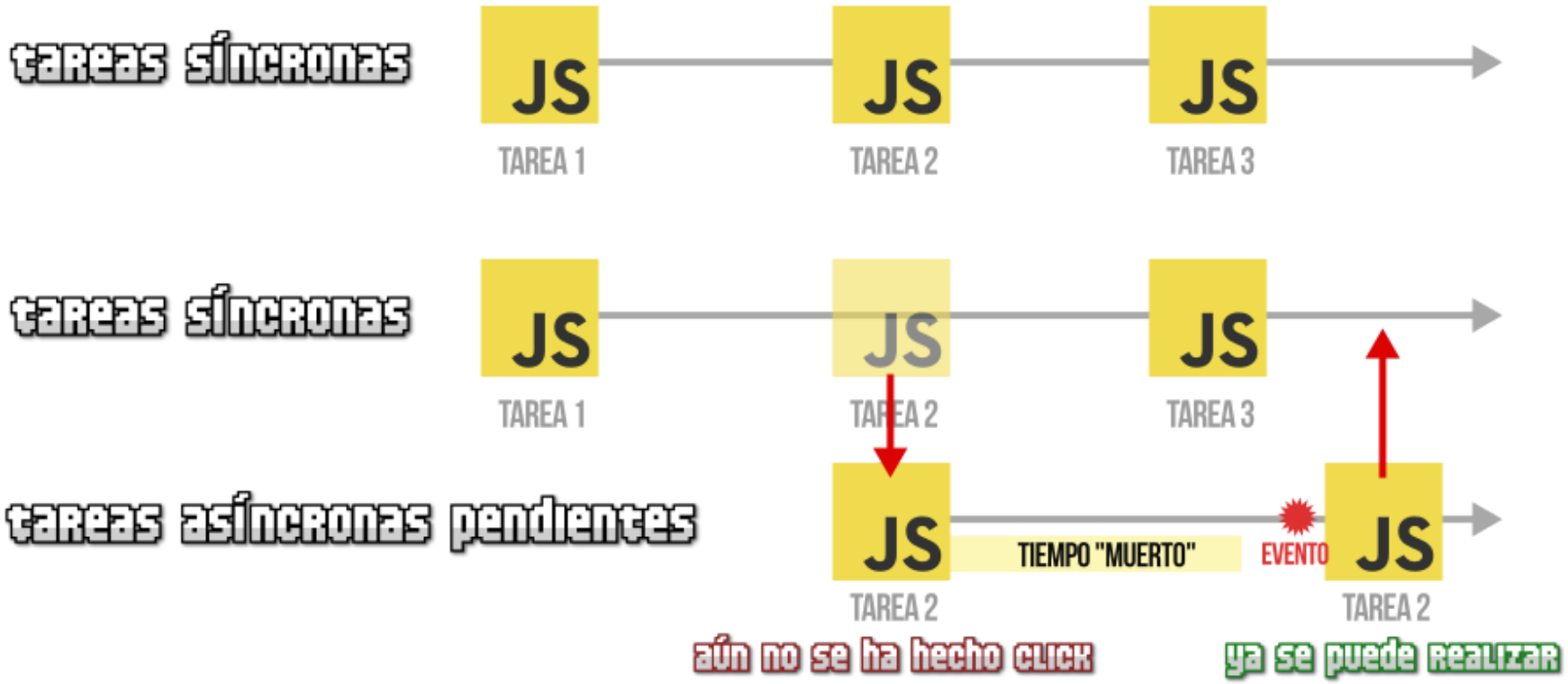
# Asincronia en JavaScript (17/10/23)

La asincronía es uno de los conceptos principales que rige el mundo de Javascript. Cuando comenzamos a programar, normalmente realizamos tareas de forma síncrona, llevando a cabo tareas secuenciales que se ejecutan una detrás de otra, de modo que el orden o flujo del programa es sencillo y fácil de observar en el código:

```
primera_funcion();    // Tarea 1: Se ejecuta primero
segunda_funcion();    // Tarea 2: Se ejecuta cuando termina primera_funcion()
tercera_funcion();    // Tarea 3: Se ejecuta cuando termina segunda_funcion()
```

Sin embargo, en el mundo de la programación, tarde o temprano necesitaremos realizar operaciones asíncronas, especialmente en ciertos lenguajes como Javascript, donde tenemos que realizar tareas que tienen que esperar a que ocurra un determinado suceso que no depende de nosotros, y reaccionar realizando otra tarea sólo cuando dicho suceso ocurra.


tareas síncronas
JS
JS
JS
TAREA 1
TAREA 2
TAREA 3
tareas síncronas
JS
JS
JS
TAREA 1
TAREA 2
TAREA 3
tareas asíncronas pendientes
JS
TIEMPO "MUERTO"
EVENTO
JS
TAREA 2
TAREA 2
aún no se ha hecho click
ya se puede realizar

Pero esto no es todo. Ten en cuenta que pueden existir múltiples tareas asíncronas, dichas tareas puede que terminen realizandose correctamente (o puede que no) y ciertas tareas pueden depender de otras, por lo que deben respetar un cierto orden.

**Ejemplos de tareas asíncronas:**

1. Un `fetch()` a una URL para obtener un archivo `.json`.
2. Un `play()` de un `.mp3` que creamos mediante un `new Audio()`.
3. Una tarea programada con `setTimeout()` que se ejecutará en el futuro.
4. Una comunicación desde Javascript a la API del sintetizador de voz del navegador.
5. Una comunicación desde Javascript a la API de un sensor del smartphone.

En Javascript existen varias formas de gestionar la asincronía, donde quizás las más populares son las siguientes