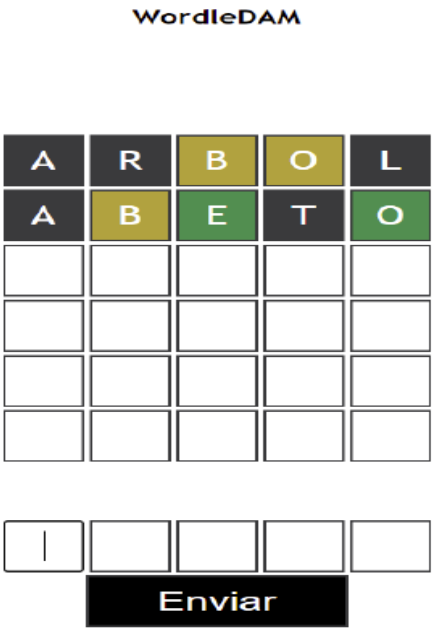
Práctica JavaScript - Wordle

Descarga la plantilla ***ejercicio1*** y descomprímela en una carpeta. Dentro verás la siguiente estructura:

- Carpeta ***css*** con archivo ***estilos.css***
- Carpeta ***js*** con archivo ***app.js***
- Página ***index.html***

Vamos a implementar una versión simplificada del juego Wordle. Este es un juego en el que hay que adivinar una palabra de X letras, con un máximo de 6 intentos. En cada intento, el programa nos dice qué letras hemos acertado, a partir de un código de tres colores:

- Las letras sombreadas en GRIS indican que NO forman parte de la palabra a descubrir
- Las letras sombreadas en AMARILLO indican que sí forman parte de la palabra, pero están en otra posición distinta
- Las letras sombreadas en VERDE indican que sí forman parte de la palabra, y además en esa misma posición



En el ejemplo de la imagen (para palabras de 5 caracteres), en el primer intento, las letras A, R y L no forman parte de la palabra a descubrir, y las letras B y O sí, pero en otras casillas. En el segundo intento, las letras E y O están en el lugar correcto, y la letra B sigue estando en un lugar equivocado.

En el archivo ***index.html*** que se proporciona, verás distintas secciones a tener en cuenta. Pasamos a explicar lo que debes hacer en cada una de ellas.

Encapsulado del código

Se pide que todo el código del archivo ***app.js*** esté encapsulado y se cargue cuando se cargue la página (evento load).

Generación de palabra secreta

Para gestionar las palabras del juego, vamos a definir un array con una serie de palabras predefinidas en mayúscula. Por ejemplo, para una palabra de longitud

5, puedes añadir estas:

PERRO, BUENO, CREMA, ARBOL, CEBRA, BRISA

Después, deberás guardar en una variable una de estas palabras seleccionada al azar. Para hacer esto, puedes usar la función *Math.random()*, que genera un número aleatorio entre 0 y 1 (sin contar el 1), combinada con *Math.floor()* para quedarte con la parte entera. Por ejemplo, el siguiente fragmento de código genera un número entero aleatorio entre 0 y N (sin contar N):

```
let numero = Math.floor(Math.random() * N);
```

Puedes adaptarlo para elegir una palabra al azar de entre las disponibles en el array, y guardarla en una variable.

Carga de la rejilla

Cuando cargues la página ***index.html*** verás que la rejilla de 6 filas no está cargada. Deberás generarla dinámicamente con JavaScript. Para ello, implementa una función llamada ***generarRejilla*** que la genere dentro de la sección con ***id="casillas"***. En el propio ***index.html*** se ha dejado comentado cómo puede ser una fila de esa rejilla. Es importante que cada una de las casillas tengan clase "casilla" para que tomen la forma cuadrada de la imagen superior, según el CSS. Así que se trata de generar 6 filas con un aspecto como este, desde JavaScript:

```
<div id="fila1">
    <div id="casilla_1_1" class="casilla"></div>
    <div id="casilla_2_1" class="casilla"></div>
    <div id="casilla_3_1" class="casilla"></div>
    <div id="casilla_4_1" class="casilla"></div>
    <div id="casilla_5_1" class="casilla"></div>
</div>
```

No es obligatorio asignar identificadores a las filas y las casillas, aunque luego te pueden facilitar su localización para comprobar las letras de las respuestas del usuario.

El botón de "Enviar"

Al pulsar el botón Enviar del formulario inferior se tendrán que desarrollar una serie de pasos que detallamos a continuación, cada uno con su puntuación correspondiente. Se valorará utilizar funciones auxiliares para resolver cada una de estas tareas, y no agruparlo todo en una sola función.

1. *Validación de las letras*: deberemos comprobar que las 5 letras que haya escritas sean efectivamente letras alfabéticas de la A a la Z (no importa si en mayúsculas o minúsculas). Si hay alguna casilla vacía o con un carácter no alfabético (por ejemplo, un dígito, o un signo de puntuación),

se mostrará un alert indicando que la palabra no es correcta.

2. *Ubicación de la palabra*: si las letras del formulario son correctas, se pasarán a mayúsculas y se añadirán en la primera fila libre de la rejilla anterior. Adicionalmente, se puede crear una variable de tipo texto con la palabra formada por esas letras, si se cree conveniente.
3. *Comprobación de coincidencias*: en este paso comprobaremos cada una de las 5 letras de la palabra introducida, para ver si coinciden o no con las de la palabra secreta. Marcaremos (usando la propiedad style en JavaScript) con color blanco el texto de cada letra, y un color de fondo diferente dependiendo de si forma parte de la palabra o no. Recuerda:
    - Color de fondo GRIS (por ejemplo, "#3a3a3c"), si la letra no forma parte de la palabra secreta
    - Color de fondo AMARILLO (por ejemplo, "#b59f3a"), si la letra sí forma parte de la palabra secreta, pero está en otra posición
    - Color de fondo VERDE (por ejemplo, "#538d4e"), si la letra sí forma parte de la palabra secreta y además está en la misma posición
4. *Fin de juego*: si hemos acertado la palabra, deberemos mostrar el texto “Has ganado” en la sección con **id=”mensaje”**. Si hemos agotado los 6 intentos, deberemos mostrar el texto “Has perdido”, seguido de la palabra secreta, en esa misma sección. En cualquiera de estos dos casos, se debe ocultar también el formulario entero (los 5 cuadros de texto y el botón).
5. *Vaciar inputs*: finalmente, vaciaremos el contenido de los 5 inputs de texto para poder introducir una nueva palabra

Evento sobre los inputs

Para finalizar, añade un evento de teclado (keyup, por ejemplo) sobre cada input del formulario inferior para que, tan pronto como escribamos una letra, pase el control al input siguiente (salvo en el último, que no avanzará más). Puedes usar para ello el método ***focus()*** de cada input, para reclamar el foco.

Mejora de la práctica

Permitir al usuario elegir entre palabras de más longitud, por ejemplo, palabras de longitud 3, 6, 10,…