CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

EXAMEN 2ª EVALUACIÓN

Desarrollo Web en entorno cliente
CFGS DAW

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2024/2025

Versión:250218.1043

Licencia

CONTENIDO

Examen 2ª evaluación – Turno Tardes

Revisa los criterios de evaluación para saber que es lo que debes tener en cuenta a la hora de realizar el ejercicio: **no es suficiente que el programa cumpla la función**, debe cumplir además otros criterios para ser considerado un buen código JavaScript. Recuerda leer bien lo que pide la práctica: no se trata de que lo hagas como pienses que debe funcionar sino como se especifica en la misma.

Puedes encontrar las plantillas de código en: https://github.com/CEED-2024/examen-segunda-evaluacion-turno-b

1. Ejercicio 1 (5 puntos)

Debes entregar un único fichero que contenga las tres funciones pedidas.

La librería `fruits.js` exporta dos funciones:

- `banana`: devuelve una promesa que se resuelve con un string
- `papaya`: devuelve una promesa que se resuelve con un string

Estas funciones pueden fallar aleatoriamente.

1.1 Apartado 1 (1 punto)

Crea una función llamada `timedExecution()` que reciba una función asíncrona como parámetro y devuelva un objeto con la siguiente estructura:

```
{
  result: "banana 45" // resultado de la función asíncrona,
  time: 12.2323 // tiempo de ejecución
}
```

No debes controlar si la función asíncrona lanza un error. Puedes obtener el tiempo actual en milisegundos con `performance.now()`.

¿Cómo llamarías a la función para ver cuánto tardas en ejecutar la función "banana"?

1.2 Apartado 2 (1 punto)

Crea una función llamada `quickFruits()` que tenga como primer parámetro el nombre de la fruta (`“banana”` o `“papaya”`) y como segundo parámetro un número natural.

La función debe lanzar el número de veces indicado la función correspondiente a la fruta en paralelo y devolver una promesa que se resuelva con un array de los resultados y del tiempo que han tardado, utilizando la función anterior (`timedExecution`). El objeto devuelto debe tener esta estructura:

```
[
  { fruit: 'banana 109', time: 100 },
  { fruit: 'banana 12', time: 300 },
  { fruit: 'banana 32', time: 130 },
]
```

Si no has implementado `timedExecution()` puedes asumir que la función ya existe.

1.3 Apartado 3 (3 PUNTOS)

Crea una función llamada `fruitRace()`. Debe lanzar en paralelo una llamada a `quickFruits` con `'banana'` y `5` y otra con `'papaya'` y `5`.

Debe devolver el resultado de las frutas más rápidas. Esto es, si la primera llamada a `quickFruits` se resuelve antes que la segunda, devolverá el array con las "bananas" y sus tiempos. Si la segunda llamada se resuelve antes, devolverá las "papayas" y sus tiempos. Debe devolver el resultado lo antes posible, en cuanto se haya recibido la primera respuesta de una de las llamadas.

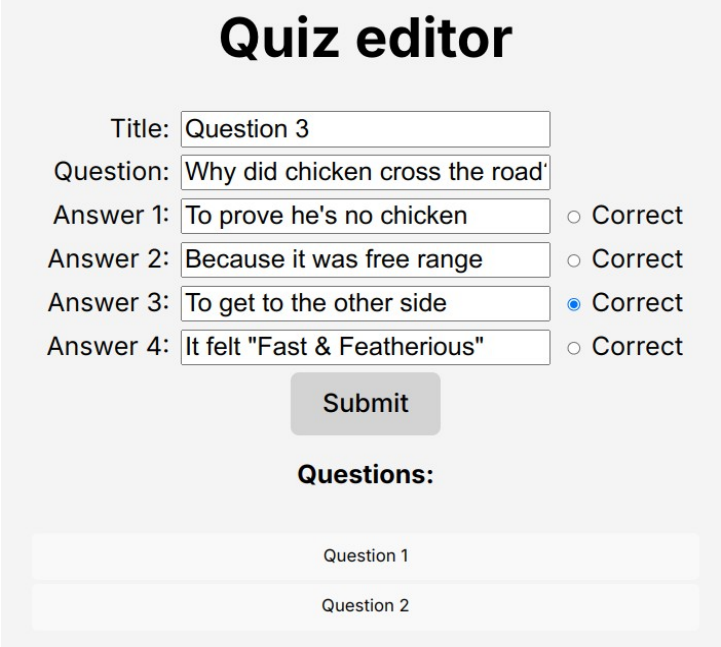
Si ambas llamadas tienen error, la función lanzará la excepción `"¡Piña!"`. Si sólo una de las llamadas tiene error, devolverá el resultado de la otra.

2. Ejercicio 2 (5 puntos)

Este ejercicio lo puedes hacer con React o con Redux Toolkit. Si lo haces con React, la nota máxima será de 3 puntos.

Entrega un fichero comprimido que contenga todo el código del proyecto, excluyendo `node_modules`.

Debes crear un editor de cuestionarios utilizando React. El aspecto debe ser el siguiente:



Debes generar un HTML similar al de la plantilla utilizada. Deben haber al menos tres componentes diferentes, pero puedes crear todos los que necesites.

El funcionamiento del juego será el siguiente:

1. Se mostrará el formulario vacío sin nada en la lista `"Questions"`
2. Al pulsar el botón, se añadirá el título de la pregunta en la lista de abajo. Se deben almacenar todos los datos de la pregunta, no sólo el título, ya que en el futuro se quiere poder editar las preguntas.
3. Si alguno de los campos está vacío o no se ha seleccionado la pregunta correcta, el botón `submit` no hará nada.

Tienes un .gif con un ejemplo de funcionamiento en las plantillas del código.

3. Linter

Cada error del linter restará tres puntos.

Las reglas del linter son las que están especificadas en `eslint.config.js` en el repositorio con las

plantillas del código, en el apartado correspondiente. No puede modificarse este fichero. Asimismo no se admitirá deshabilitar ninguna de las reglas de eslint por ningún medio: en ese caso se procederá como si el programa hubiese fallado el linter, y se restarán tres puntos por cada error detectado sin ese cambio.

4. Criterios de evaluación

- El programa es correcto, realiza la función que se solicita en el enunciado
- Se han utilizado estructuras del lenguaje adecuadas: bucles, condicionales, operadores, etc.
- Se ha estructurado correctamente el código utilizando módulos
- Se han utilizado variables y constantes de forma adecuada
- Se utilizan correctamente y cuando corresponda los tipos de datos y objetos predefinidos del lenguaje (Arrays, objetos planos, Map, Set, etc.)
- Se han utilizado funciones para estructurar el código, definiendo y utilizando parámetros y valores de respuesta de forma adecuada
- El programa es lo más sencillo posible para realizar su función.
- No existe código repetido: se han extraído los comportamientos comunes a funciones y se ha intentado hacer el código genérico.
- El programa cumple todas las reglas definidas para el linter.
- Se han utilizado correctamente las funciones de React
- La distribución de la aplicación en componentes es adecuada.
- Se ha programado la aplicación utilizando React y Redux toolkit
- El diseño de los stores, actions y reducers es correcto.
- Se utilizan thunks asíncronos apropiadamente.
- El programa cumple todas las reglas definidas para el linter.
- Se genera el HTML solicitado