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

2022/2023

Versión:250218.0950

Licencia

CONTENIDO

Examen 2ª evaluación – Turno Mañanas

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-a

1. Ejercicio 1 (5 puntos)

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

La libreria `fruits.js` exporta tres funciones:

- `fruit`: devuelve una promesa que se resuelve con un string que contiene `'banana'` o `'papaya'` de forma aleatoria
- `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 `getFruit()` que llame a `fruit` y, en función de la fruta devuelta (`'banana'` o `'papaya'`) llame a `banana()` o `papaya()` respectivamente.

La función debe devolver una promesa que se resuelva con el resultado de llamar a `banana()` o `papaya()`. No hace falta que controles los errores de `fruit` en la función.

¿Cómo llamarías a la función para obtener el resultado de la promesa?

1.2 Apartado 2 (1 punto)

Crea una función llamada `fiveFruits` que llame a `getFruit()` 5 veces y devuelva una promesa que se resuelva con un array de los resultados.

Las llamadas a `getFruit()` deben hacerse en paralelo. No hace falta que controles los errores de `getFruit` en esta función.

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

1.3 Apartado 3 (3 puntos)

Crea una función llamada `tenFruits`. Debe lanzar en paralelo 3 llamadas a `fiveFruits` e imprimir mediante `console.log` las diez primeras frutas que obtenga. Las puede imprimir tan pronto como las obtenga, no es necesario esperar a obtener las diez frutas para empezar a imprimir.

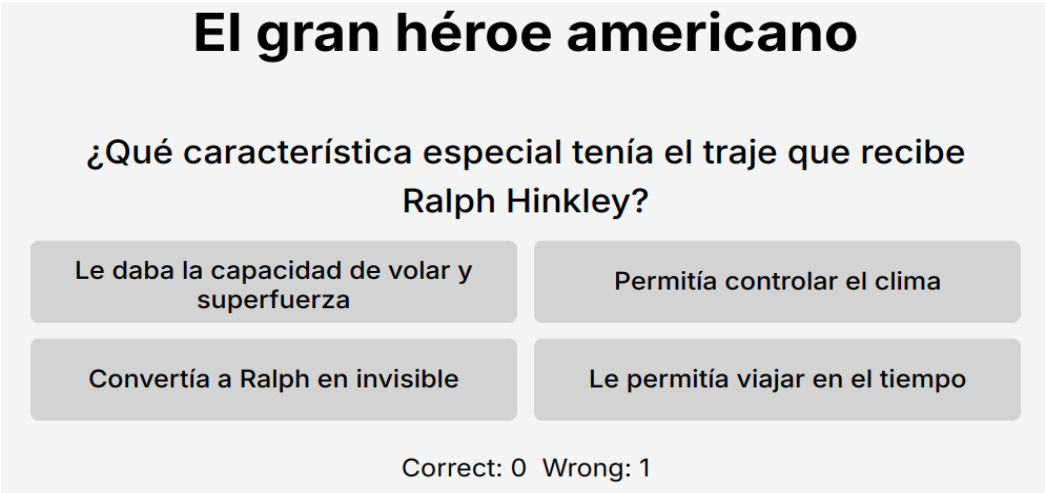
Si alguna de las llamadas a `fiveFruits` falla, la función lanzará la excepción `"Mango!"`.

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 juego de preguntas y respuestas 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 cargarán las preguntas importando `quiz.js`.
2. Se mostrará la primera pregunta. Al seleccionarla, se incrementará el contador de respuestas correctas o incorrectas, y se pasará a la siguiente pregunta.
3. La última pregunta sólo se puede responder una vez. Una vez respondida el juego permanecerá en esa pregunta mostrando el número total de respuestas correctas y erróneas.
4. Ten en cuenta que el programa debe funcionar si se utiliza con otro fichero `quiz.js` que contenga un contenido diferente. El número de posibles respuestas para cada pregunta podría variar, así como el número de preguntas.

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