09. PROMISES JAVASCRIPT

IES ESTACIÓ CURS 2021- 2022

Índex

- 1. Promises.
- 2. Exemples.

1. Promises.

- Des de l'especificació ES6 o ES2015 de JavaScript, les promeses ja són natives i no necessitem requerir a llibreries de tercer.

- Utilitzarem l'exemple de **addToArray,** anteriorment vist en **Callbacks**:

```
function addToArray(data, array) {
  const promise = new Promise(function (resolve, reject) {
    setTimeout(function () {
      array.push(data);
      resolve(array);
    }, 1000);

    if (!array) {
      reject(new Error("No existe un array"));
    }
  });

  return promise;
}

const array = [1, 2, 3];
addToArray(4, array).then(function () {
  console.log(array);
});
```

1. Promises.

- Ara la funció **addToArray** crea un objecte **Promise** que rep com a paràmetres una funció amb les funcions **resolve** i **reject**.
- **resolve** la cridarem quan la nostra execució finalitze correctament.
- D'aquesta manera, podem escriure codi de manera més elegant, i el **Callback Hell** anterior pot ser resolt així:
- Açò és coneix com "anidar promeses".

```
const array = [1, 2, 3];
addToArray(4, array)
  .then(function () {
    return addToArray(5, array);
  })
  .then(function () {
    return addToArray(6, array);
  })
  .then(function () {
    return addToArray(7, array);
  })
  .then(function () {
    console.log(array);
  });

// (4 seg. de delay)-> [1,2,3,4,5,6,7]
```

1. Promises.

- La manera de tractar errors en una promesa, és per mitjà de la funció **catch** que recull el que enviem en la funció **reject** dins de la Promesa.

- I aquesta funció només cal invocar-la una vegada, no necessitem comprovar en cada crida si existeix error o no.
- Açò redueix molt la quantitat de codi

```
const array = ''
addToArray(4, array)
  .then(...)
  .then(...)
  .then(...)
  .catch(err => console.log(err.message))

// No existe el array
```

2. Exemples.

- **Construeix un joc d'endevinalles.**

  - Els requeriments:

  - Un usuari pot introduir un número.
  - El sistema tria un número aleatori de l'1 al 6.
  - Si el número d'usuari és igual al número aleatori, li dóna a l'usuari 2 punts.
  - Si el número de l'usuari és diferent al número aleatori per 1, li dóna a l'usuari 1 punt. D'una altra manera, li dóna a l'usuari 0 punts.
  - L'usuari pot jugar tant com vulga.

2. Exemples.

- Crearem una funció **insertaNum** i retorna una **Promise**:

```
const insertaNum = () => {
  return new Promise((resolve, reject) => {
    // Empecemos desde aquí...
  });
};
```

2. Exemples.

- Primer hem de demanar-li un número a l'usuari i triar un número aleatori entre 1 i 6:

```
const insertaNum = () => {
  return new Promise((resolve, reject) => {
    const numUsuario = Number(window.prompt("Introduce un número (1 - 6):"));
      // Pide al usuario que introduzca un número

    const aleatorio = Math.floor(Math.random() * 6 + 1);
      // Elige un número aleatorio del 1 al 6
  });
};
```

2. Exemples.

- Ara, **numUsuario** pot ingressar un valor que no és un número.
- Si és així, cridem a la funció **reject** amb un error:

```
const insertaNum = () => {
  return new Promise((resolve, reject) => {
    const numUsuario = Number(window.prompt("Introduce un número (1 - 6):"));
      // Pide al usuario que introduzca un número

    const aleatorio = Math.floor(Math.random() * 6 + 1);
      // Elige un número aleatorio del 1 al 6

    if (isNaN(numUsuario)) {
      reject(new Error("Tipo de entrada incorrecta"));
        // Si el usuario introduce un valor que no es un número,
        // ejecuta reject con un error
    }
  });
};
```

2. Exemples.

- El següent que volem fer és verificar si el **numUsuario** és igual a aleatori, si és així, volem donar-li a l'usuari 2 punts i podem executar la funció resolve passant un **objecte** {punts: 2, aleatori}.
- Observa que també volem saber el número en aleatori quan és resolga la **promesa**.
- Si ell **numUsuario** és diferent d'aleatori per un, aleshores li donem a l'usuari 1 punt.
- En cas contrari, li donem a l'usuari 0 punts:

2. Exemples.

```
return new Promise((resolve, reject) => {
  const numUsuario = Number(window.prompt("Introduce un número (1 - 6):"));
    // Pide al usuario que introduzca un número
  const aleatorio = Math.floor(Math.random() * 6 + 1);
    // Elige un número aleatorio del 1 al 6

  if (isNaN(numUsuario)) {
    reject(new Error("Tipo de entrada incorrecta"));
        // Si el usuario introduce un valor que no es un número,
        // ejecuta reject con un error
  }

  if (numUsuario === aleatorio) {
    // Si el número del usuario coincide con el número aleatorio,
    // retorna 2 puntos
    resolve({
      puntos: 2,
      aleatorio,
    });
  } else if (numUsuario === aleatorio - 1 || numUsuario === aleatorio + 1) {
    // Si el número del usuario es diferente al número aleatorio por 1,
    // retorna 1 punto
    resolve({
      puntos: 1,
      aleatorio,
    });
  } else {
    // Si no, retorna 0 puntos
    resolve({
      puntos: 0,
      aleatorio,
    });
  }
});
```

2. Exemples.

- També creiem una altra funció per a preguntar si l'usuari vol continuar el joc:
- Observa que creguem una **Promise**, però no utilitza la funció **callback reject**. Això està totalment bé.

```
const continuarJuego = () => {
  return new Promise((resolve) => {
    if (window.confirm("¿Quieres continuar?")) {
        // Pregunta si el usuario quiere continuar el juego
        // con un modal de confirmación
      resolve(true);
    } else {
      resolve(false);
    }
  });
};
```

2. Exemples.

- Ara creiem una funció per a manejar la suposició:

```
const suponer = () => {
  insertaNum() // Esto retorna una Promesa
    .then((result) => {
      alert(`Dado: ${result.aleatorio}: obtuviste ${result.puntos} puntos`);
      // Cuando resolve se ejecuta, obtenemos los puntos
      // y el número aleatorio

      // Vamos a preguntarle al usuario si quiere continuar el juego
      continuarJuego()
          .then((result) => {
                if (result) {
                  suponer(); // Si sí, ejecutamos suponer() de nuevo
                } else {
                  alert("Terminó el juego"); // Si no, mostramos una alerta
                }
          });
    })
    .catch((error) => alert(error));
};

suponer(); // Ejecuta la función suponer.
```

2. Exemples.

- Ací quan anomenem **suposar()**, **insertaNum()** ara retorna una **Promise**:

- Si la **Promise** és resolta, cridem al mètode **then** i vam mostrar un missatge d'alerta.
- També preguntem si l'usuari vol continuar.
- Si la **Promise** és rebutjada, vam mostrar un missatge d'alerta amb l'error.
- Com pots veure, el codi és una cosa difícil de llegir.

- Refactoritzem la funció **suponer** una mica, utilitzant la sintáxis **async/await**:

2. Exemples.

```
const suponer = async () => {
  try {
    const result = await insertaNum();
      // En lugar del método 'then', podemos obtener el resultado
      // directamente, poniendo 'await' antes de la promesa
    alert(`Dado: ${result.aleatorio}: obtuviste ${result.puntos} puntos`);

    const estaContinuando = await continuarJuego();

    if (estaContinuando) {
      suponer();
    } else {
      alert("Terminó el juego");
    }
  } catch (error) {
      // En lugar del método 'catch', podemos usar la sintáxis 'try/catch'
    alert(error);
  }
};
```

2. Exemples.

- Pots veure que creguem una funció **async**, col·locant **async** abans de les claus. Aleshores en la funció **async**:
  - En lloc del mètode **then**, podem obtindre els resultats directament, posant **await** abans de la promesa.
  - En lloc del mètode **catch**, podem utilitzar la sintáxis **try/catch**.
- Ací està tot el codi per a aquesta tasca, de nou, per a la vostra referència:

2. Exemples.

```
return new Promise((resolve, reject) => {
  const numUsuario = Number(window.prompt("Introduce un número (1 - 6):"));
    // Pide al usuario que introduzca un número
  const aleatorio = Math.floor(Math.random() * 6 + 1);
    // Elige un número aleatorio del 1 al 6

  if (isNaN(numUsuario)) {
    reject(new Error("Tipo de entrada incorrecta"));
        // Si el usuario introduce un valor que no es un número,
        // ejecuta reject con un error
  }

  if (numUsuario === aleatorio) {
    // Si el número del usuario coincide con el número aleatorio,
    // devuelve 2 puntos
    resolve({
      puntos: 2,
      aleatorio,
    });
  } else if (numUsuario === aleatorio - 1 || numUsuario === aleatorio + 1) {
    // Si el número del usuario es diferente al número aleatorio por 1,
    // devuelve 1 punto
    resolve({
      puntos: 1,
      aleatorio,
    });
  } else {
    // Si no, devuelve 0 puntos
    resolve({
      puntos: 0,
      aleatorio,
    });
  }
});
```

2. Exemples.

```
const continuarJuego = () => {
  return new Promise((resolve) => {
    if (window.confirm("¿Quieres continuar?")) {
        // Pregunta si el usuario quiere continuar el juego
        // con un modal de confirmación
      resolve(true);
    } else {
      resolve(false);
    }
  });
};


const suponer = async () => {
  try {
    const result = await insertaNum();
      // En lugar del método 'then', podemos obtener el resultado
      // directamente, poniendo 'await' antes de la promesa
    alert(`Dado: ${result.aleatorio}: obtuviste ${result.puntos} puntos`);

    const estaContinuando = await continuarJuego();

    if (estaContinuando) {
      suponer();
    } else {
      alert("Terminó el juego");
    }
  } catch (error) {
      // En lugar del método 'catch', podemos usar la sintaxis 'try/catch'
    alert(error);
  }
};
```