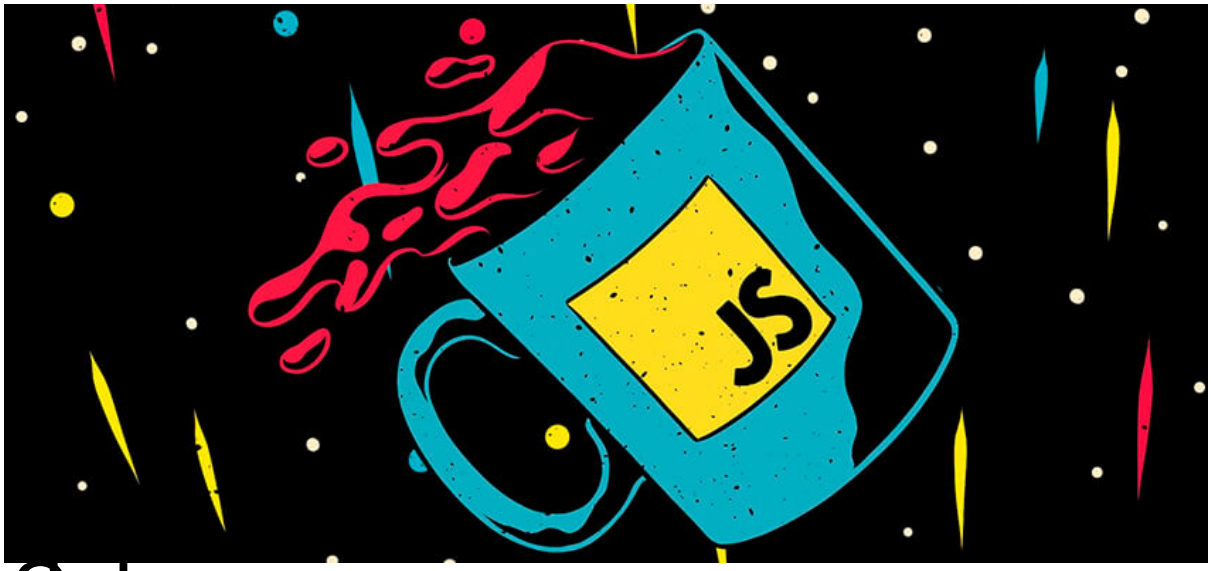

Javascript S8 | Tratamiento de errores y destructuring

Después de esta lección podrás:

1. Manejo de errores dentro de una aplicación Js.
2. Control del flujo de errores en una aplicación Js.
3. Destructuring Js avanzado.

Tratamiento de errores

En JavaScript, al igual que en otros lenguajes, los errores se tratan mediante excepciones. Cuando
la ejecución de una línea produce un error, esta lanza una excepción.

Las excepciones se propagan de manera ascendente desde la última función ejecutada, ascendiendo hasta la primera y esperando ser capturada en algún punto.

Para **capturar** una **excepción** debemos **ejecutar el código dentro de la sentencia try catch**. La declaración try...catch marca un bloque de instrucciones para **intentar (try)**, y especifica una respuesta si se produce una **excepción (catch)**.

```
try {
  // Definimos un JSON erroneo
  let jsonErroneo = "{ var: 123123, hola: ";
  // Tratamos de convertirlo a objeto (fallará)
  let json = JSON.parse(jsonErroneo);
  // Si no ha fallado nada deberíamos imprimir esta línea
  console.log("Hemos conseguido terminar el bloque try !!");
} catch (e) {
  console.error("No se ha podido ejecutar el bloque try !!");
  console.error("Excepcion:");
  console.error(e);
}
```

> Somos conscientes de que el bloque del try es posible que falle, por eso lo ejecutamos dentro de try catch. En este

> caso JSON.parse fallará y lanzará una excepción que será recogida por el catch.

Lanzando Excepciones

Si en algún momento durante la ejecución de nuestro código quisiéramos **indicar que se ha producido un error**, podemos lanzar de manera voluntaria una excepción, que será tratada de la manera que hemos visto en el punto anterior. Para lanzar una excepción debemos ejecutar el siguiente código:

```
try {
  // Definimos un JSON erroneo
  let jsonErroneo = "{ var: 123123, hola: ";
  // Tratamos de convertirlo a objeto (fallará)
  let json = JSON.parse(jsonErroneo);
  // Si no ha fallado nada deberíamos imprimir esta línea
  console.log("Hemos conseguido terminar el bloque try !!");
} catch (e) {
  // genera una excepción con un valor cadena
  throw "Ha ocurrido un error";
}
```

> Podemos enviar cualquier valor en la excepción, desde tipos primitivos hasta objetos complejos.

spread/rest operator

Ahora el operador `...` que llegó con ES6 permite escribir código mucho más limpio y simplifica la forma en que llevamos a cabo distintas operaciones.

Por ejemplo, podemos rellenar *arrays* de la siguiente forma:

```
// Definimos un array
let idolosDelRayismoUno = ['Piti', 'Tamudo'];
// Definimos un segundo array incluyendo los valores del primero con ...
let idolosDelRayismoDos = ['Chori', ...first, 'Míchel'];

// ["Chori", "Piti", "Tamudo", "Míchel"]
```

Trabajar con objetos inmutables de una forma más sencilla:

```
// Definimos un Objeto
let rayista = { name: 'Wilfred Agbonavbare' };
// Definimos un segundo Objeto incluyendo los valores del primero con ...
let masRayista = { ...rayista, position: 'goalkeeper' };

// { name: 'Wilfred Agbonavbare' , position: 'goalkeeper' }
```

O recoger los argumentos de una función del siguiente modo:

```
// Declaramos la función y por parametro recibe dos valores y más...
function idolosRayismo(a, b, ...masJugadores) {
  console.log(a);
  console.log(b);
  console.log(masJugadores);
}

// Llamamos a mi función
idolosRayismo('Míchel', 'Cota', 'Piti', 'Michu', 'Tito');

// Nos devolverá los valores agrupados en un Array
// 'Míchel' 'Cota' ['Piti', 'Michu', 'Tito']
```

Arrow functions y destructuring

Una característica con la que las ***arrow functions*** **se llevan genial es con el *destructuring assignment***. De este modo, **cuando pasemos un objeto a una función de este tipo, podremos acceder directamente a sus propiedades de una forma muy visual**:

```
const printIdolosRayismo = ({name}) => console.log(name);

printIdolosRayismo({name: 'Wilfred Agbonavbare', position: 'goalkeeper'});
```

Así como **declarar parámetros por defecto** empleando esta característica:

```
const printIdolosRayismo = ({name} = {name: 'Piti'}) => console.log(name);

printIdolosRayismo({name: 'Wilfred Agbonavbare', position: 'goalkeeper'});

// Imprime 'Wilfred Agbonavbare'

printIdolosRayismo();

// Imprime 'Piti'
```