Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Async JavaScript

IIC2513 - Tecnologías y Aplicaciones Web

Sebastián Vicencio R.
2do Semestre 2020

Async

¿Qué significa?

Blocking code

Código se ejecuta en orden

```
console.log('Hello');

for (let i = 0; i < 1000000000; i += 1) {
}

console.log('Bye');
```

JavaScript es single-threaded

Async

¿Y si queremos ejecutar una operación que no sabemos cuánto va a demorar?

Async code

Ciertas operaciones no bloquean

```
setTimeout(function()  {
  console.log('You waited 3 seconds to see this')
}, 3000)
```

Especificamos lo que sucederá cuando se complete

Event loop

Una cola de espera

- Llamadas asíncronas se agregan a una cola
- Main thread continúa ejecutando código
- Cuando main thread ejecutó todo, elementos de la cola se comienzan a ejecutar
- Ejecución es en orden en que fueron encolados

Event loop

Un ejemplo

```
console.log('First');

setTimeout(function()  {
  console.log('You waited 3 seconds to see this')
}, 3000);

console.log('Last');
```

Event loop

¿Y esto?

```
console.log('First');

setTimeout(function()  {
  console.log('You waited 3 seconds to see this')
}, 0);

console.log('Last');
```

Más sobre event loop: https://www.youtube.com/watch?v=8aGhZQkoFbQ

Callbacks

Async tradicional

Callbacks

Función que se pasa como parámetro a otra

`forEach`

Síncronos

`setTimeout`

Asíncronos

Callbacks

Estructura general

```
asyncFunction(function(param) {
  console.log(param);
})


function asyncFunction(callback) {
  // It performs some async operation like reading a file
  const data = getAsyncDataSomehow();
  callback(data);
}
```

Callbacks

Callback puede llamar a otra función asíncrona

```
asyncFunction(function(param) {
  anotherAsyncFunction(param, function(anotherParam) {
    console.log(anotherParam);
  })
})
```

Callback Hell

Pyramid of Doom: múltiple anidación

```
asyncFunction(function(param) {
  anotherAsyncFunction(param, function(anotherParam) {
    yetAnotherAsyncFunction(anotherParam, function(yetAnotherParam) {
      lastAsyncFunction(yetAnotherParam, function(lastParam) {
        console.log(lastParam);
      })
    });
  });
});
```

Más sobre callback hell: http://callbackhell.com/

Promesas

Async moderno

Promesas

¿Qué son?

Objeto que representa un estado intermedio de una operación que se resolverá en el futuro

Es la promesa de un valor futuro.

“No puedo ahora, pero te prometo que retornaré un valor”

Promesas

Promised-based function

```
function asyncFunction() {
  return Promise.resolve('You see? I promised');
}

console.log(asyncFunction());
```

Función que retorna una promesa

Promesas

Estados y ciclo de vida

1. `pending` - Creación
2. `fulfilled` - Resuelta con éxito
3. `rejected` - Resuelta con error

Promesas

“...And then I promised”

```
asyncFunction().then(function(promisedValue) {
    console.log(promisedValue);
  });
```

`then` - Método de toda promesa

Promesas

¿Qué pasa ahora con esto?

```
console.log('First');

asyncFunction().then(function(promisedValue) {
  console.log(promisedValue);
});

console.log('Last');
```

Promesas

“But I promise again”

`then` - Retorna una promesa

- Promesa se resuelve al valor retornado en callback
- Si callback retorna promesa, se resuelve al valor al que resuelve esta promesa

```
asyncFunction()
  .then(function(promisedValue) {
    return promisedValue + ', and I\'d do it again';
  })
  .then(function(promisedValue) {
    console.log(promisedValue);
  })
```

Promesas

¿Cómo evitar callback hell?

```
asyncFunction(function(param) {
  anotherAsyncFunction(param, function(anotherParam) {
    yetAnotherAsyncFunction(anotherParam, function(yetAnotherParam) {
      lastAsyncFunction(yetAnotherParam, function(lastParam) {
        console.log(lastParam);
      })
    });
  });
});
```

Promesas

¿Cómo evitar callback hell?

```
asyncFunction()
  .then(function(param) {
    return anotherAsyncFunction(param);
  })
  .then(function(anotherParam) {
    return yetAnotherAsyncFunction(anotherParam);
  })
  .then(function(yetAnotherParam) {
    return lastAsyncFunction(yetAnotherParam);
  })
  .then(function(lastParam) {
    console.log(lastParam)
  });
```

Promesas

Una versión más corta

```
asyncFunction()
  .then(param => anotherAsyncFunction(param))
  .then(anotherParam => yetAnotherAsyncFunction(anotherParam))
  .then(yetAnotherParam => lastAsyncFunction(yetAnotherParam))
  .then(lastParam => console.log(lastParam));
```

Promesas

Un ejemplo real

```
function displayRemoteData(url) {
  return fetch(url)
    .then(response => response.json())
    .then(response => console.log(response))
    .catch(message => console.log('An error occurred:', message));
}


displayRemoteData('https://jsonplaceholder.typicode.com/users')
```

Actividad

Hands-on