

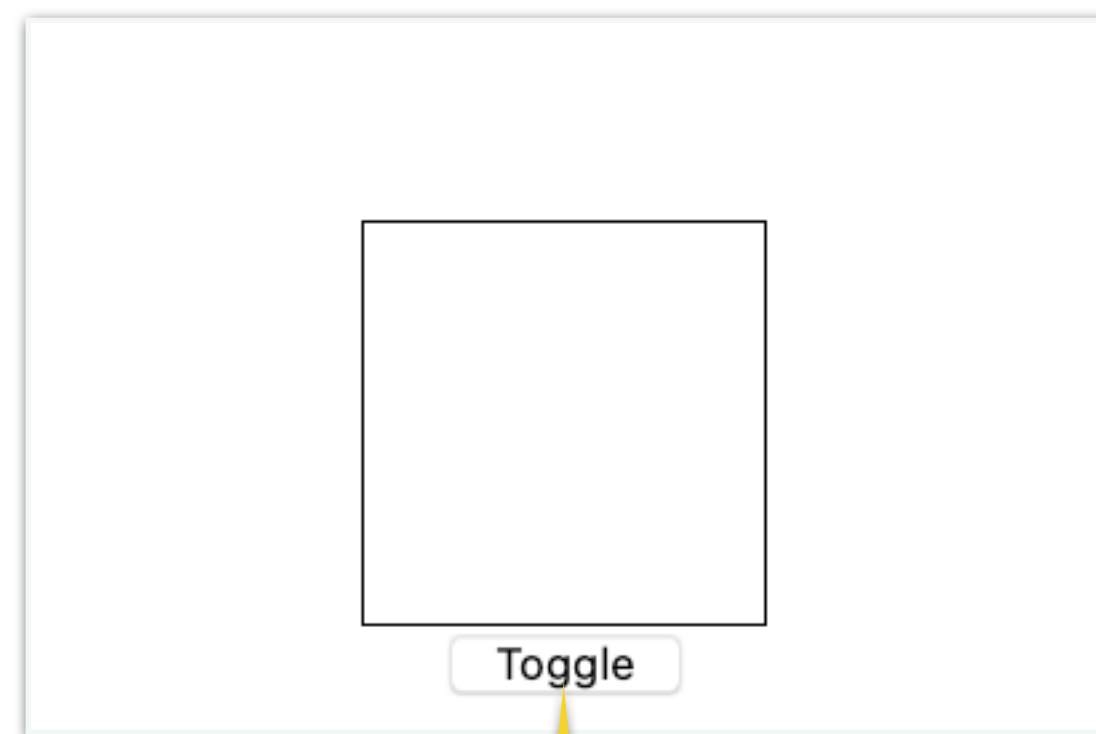
# Web Programming

# **Async JS**

**Leander Jehl** | University of Stavanger

# Example

🔗 [examples/js/async/timeout](#)



**Challenge:** Allow **Toggle** button to stop and continue color sequence.

```
function colorsequence(){
  const box = document.getElementById("box");
  // add class red
  box.classList.add("red");
  // after 2sec replace class red with blue
  setTimeout(
    function(){
      box.classList.replace("red", "blue");
    },
    2000
  );
  // after another 2sec replace class blue with green
  // after another 2sec replace class green with yellow
```

callback function

Defining callbacks makes difficult code.

# Example

🔗 [examples/js/async/stop\\_and\\_start](#)

- **await** stops the function.
- Function continues when it receives completion signal.
- Function using **await** must be marked as **async**

```
async function colorsequence(){  
  const box = document.getElementById("box");  
  // red after 2s blue, after 2s green  
  box.classList.add("red");  
  //wait for 2 sec  
  await sleep(2000);  
  // change to blue  
  box.classList.replace("red", "blue");  
  await sleep(2000);  
  ...  
}
```

Call `sleep` function.  
Continue `colorsequence`, when `sleep` signals completion.

# Promise

- `await` must receive a promise

```
await new Promise( ... );
```

- the `Promise` is created with a function

```
function(resolve){  
  // do stuff or wait  
  resolve();  
}
```

- The first argument to that function, is the resolve function. If resolve is called, the promise is resolved and the `async` function continues after the `await`.

# Example

🔗 [examples/js/async/stop\\_and\\_start](#)

## sleep function

- create Promise
- pass resolve function to setTimeout and call after timeout
- return promise

```
function sleep(ms){
  let promise = new Promise(
    function(resolve){
      setTimeout(
        function(){
          resolve()
        },
        ms
      );
    }
  );
  return promise;
}
```

```
function sleep(ms){
  return new Promise(
    (resolve)=> setTimeout(resolve, ms)
  );
}
```

# Example

🔗 [examples/js/async/stop\\_and\\_start](#)

## sleep function

- create Promise
- pass resolve function to setTimeout and call after timeout
- return promise

```
function sleep(ms){
  let promise = new Promise(
    function(resolve){
      setTimeout(
        function(){
          resolve()
        },
        ms
      );
    }
  );
  return promise;
}
```

```
function sleep(ms){
  return new Promise(
    (resolve)=> setTimeout(resolve, ms)
  );
}
```

# Example

🔗 [examples/js/async/stop\\_and\\_start](#)

- **await** takes a **Promise** and stops the function
- Function continues when the **Promise** is resolved.
- Function using **await** must be marked as **async**

```
async function colorsequence(){  
  const box = document.getElementById("box");  
  // red after 2s blue, after 2s green  
  box.classList.add("red");  
  //wait for 2 sec  
  await sleep(2000);  
  // change to blue  
  box.classList.replace("red", "blue");  
  await sleep(2000);  
  ...  
}
```

Call `sleep` function.  
Continue `colorsequence`, when `sleep` signals completion.

# Exercise #1



[github.com/dat310-spring21/course-info/tree/master/](https://github.com/dat310-spring21/course-info/tree/master/exercises/async/js)  
**exercises/async/js**



# Promise can return a value

- resolve can receive a value

```
function(resolve){  
  // do stuff or wait  
  resolve("red");  
}
```

- Value is available when promise is resolved

```
let color = await getNext();
```

# Promise can raise an error

- reject will cause an error

```
function doStuff(){  
  return new Promise(  
    (resolve, reject) => {  
      // do stuff  
      if (ok) resolve();  
      else reject();  
    }  
  )  
}
```

- await inside try, catch

```
try {  
  await doStuff();  
} catch {  
  console.log("An error was raised");  
}
```

# Example

🔗 [examples/js/async/forever](#)

```
async function colorsequence(){
  const box = document.getElementById("box");
  box.classList.add("red");
  try {
    while (true){
      let color = await getNext();
      console.log("Change to ", color)
      box.classList.remove("red", "blue", "green", "yellow");
      box.classList.add(color);
    }
  }
  catch(err){
    console.log(err);
  }
  finally {
    box.classList.remove("red", "blue");
  }
}
```

# Exercise #2



[github.com/dat310-spring21/course-info/tree/master/](https://github.com/dat310-spring21/course-info/tree/master/exercises/async/js)  
**exercises/async/js**