

# Async/Await

---

*I Promise to await for async code...*



# What is “asynchronous” code?

*Asynchronous* (aka *async*) literally means "happening at disconnected times."

Async code in JS will run at an arbitrary (unknown) future time, and *other JS code* can run in the *meantime*.

# Operations that run Asynchronously

- ◉ A few examples include:
  - Opening, reading, and writing [to] files
  - **Making API calls** such as HTTP requests from our frontend
  - **Accessing a database** (e.g. create, read, update, delete)



# What is “asynchronous” code?

```
console.log("One")  
  
setTimeout(() => console.log("Two"), 10)  
  
console.log("Three")
```

- In which order will the logs fire?



# What is “asynchronous” code?

```
console.log("One")  
  
setTimeout(() => console.log("Two"), 10)  
  
console.log("Three")
```

- In which order will the logs fire?

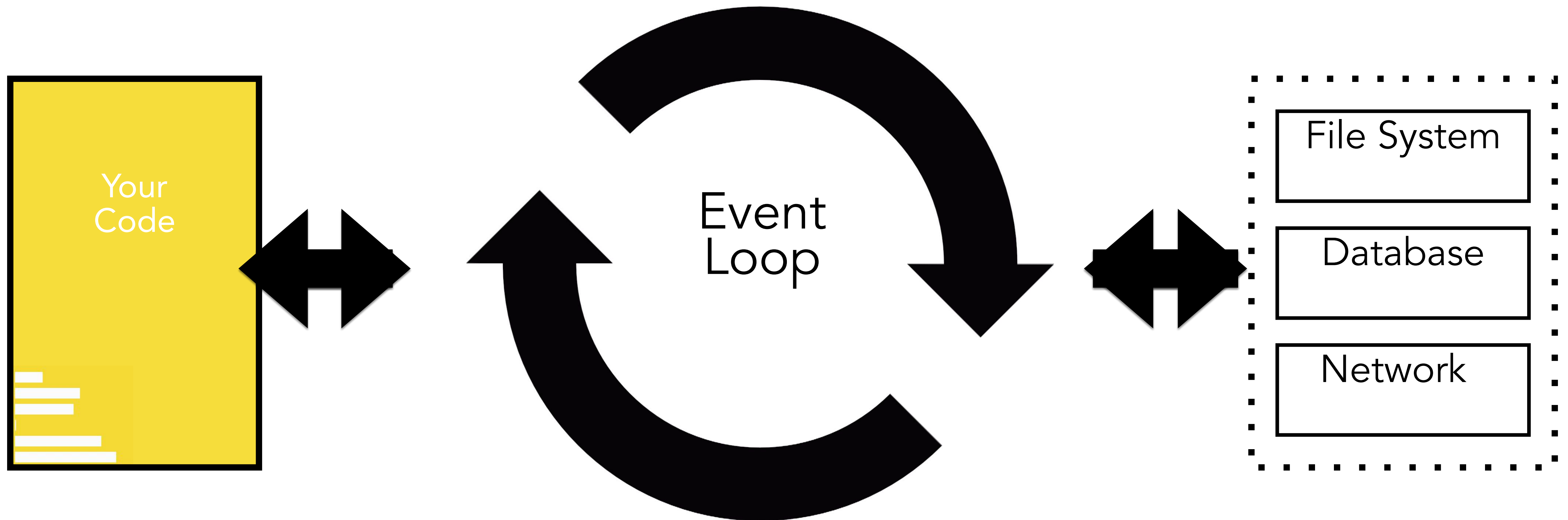
One

Three

Two



# What is “asynchronous” code?



# How to handle asynchronous code?

## 1. Callbacks



# Async with callbacks

```
console.log("Getting Configuration")

fs.readFile('/config.json', 'utf8', (err, data) => {

  console.log("Got configuration:", data)

});

console.log("Moving on...");
```

- ◉ BTW, in which order will the logs fire?





# Problems with callbacks

```
const tryGetRich = () => {  
  readFile('/luckyNumbers.txt', (err, fileContent) => {  
    // Do something with lucky numbers  
  })  
}
```



# Problems with callbacks

```
const tryGetRich = () => {  
  readFile('/luckyNumbers.txt', (err, fileContent) => {  
    nums = fileContent.split(",");  
    nums.forEach(num => {  
      bookmaker.getHorse(num, (err, horse) => {  
        // Ok, this is getting a little confusing  
      })  
    })  
  })  
}
```



# Problems with callbacks

```
const tryGetRich = () => {  
  readFile('/luckyNumbers.txt', (err, fileContent) => {  
    nums = fileContent.split(",");  
    nums.forEach(num => {  
      bookmaker.getHorse(num, (err, horse) => {  
        bookmaker.bet(horse, (err, success) => {  
          (success) {  
            // Help.  
          }  
        })  
      })  
    })  
    console.log('When will I run??')  
  })  
})  
}
```

**CALLBACK HELL**



# How to handle asynchronous code?

1. Callbacks
2. Promises



# Callbacks vs Promises

## CALLBACKS

```
const tryGetRich = () => {  
  
  readFile('/luckyNumber.txt', (err, num) => {  
    bookmaker.bet(num, (err, success) => {  
      if(success) {  
        console.log("I'm rich!")  
      }  
    })  
  })  
}
```



# Callbacks vs Promises

## CALLBACKS

```
const tryGetRich = () => {  
  
  readFile('/luckyNumber.txt', (err, num) => {  
    bookmaker.bet(num, (err, success) => {  
      if(success) {  
        console.log("I'm rich!")  
      }  
    })  
  })  
}
```

## ASYNC/AWAIT (PROMISES)

```
const tryGetRich = async () => {  
  
  let num = await readFileAsync('/luckyNumber.txt')  
  let success = await bookmaker.bet(num)  
  
  if(success) {  
    console.log("I'm rich!")  
  }  
  
}
```

*...but we are getting ahead of ourselves.*

# What is a Promise?

- ◉ A Promise is a JavaScript object that represents the eventual result of an asynchronous operation.
- ◉ Again, just an object with *value* and *status*.





# What is a Promise?

```
readFileAsync('/luckyNumber.txt')
```

```
{  
  [[PromiseValue]]: undefined,  
  [[PromiseStatus]]: "pending"  
}
```





# What is a Promise?

```
readFileAsync('/luckyNumber.txt')
```

```
{  
  [[PromiseValue]]: "42",  
  [[PromiseStatus]]: "fulfilled"  
}
```



# Promise

```
const num = readFileAsync('/luckyNumber.txt')
```



# async/await

```
const num = await readFileAsync('/luckyNumber.txt')
```



# async/await

```
async function getNumber() {  
  const num = await readFileAsync('/luckyNumber.txt')  
}  
  
getNumber()
```



# async/await

```
const getNumber = async () => {  
  const num = await readFileAsync('/luckyNumber.txt')  
}  
  
getNumber()
```

*Demo*

# A word on error handling...





# Try/Catch

```
function getLuckyGem(birthMonth) {  
  const gems = ['Emerald', 'Amethyst', 'Jade', 'Opal', 'Sapphire', 'Perl',  
                'Ruby', 'Agate', 'Diamond', 'Moonstone', 'Jasper', 'Onyx'];  
  if (gems[birthMonth]) {  
    return gems[birthMonth];  
  } else {  
    throw new Error('Invalid birth Month');  
  }  
}  
  
try { // statements to try  
  myGem = getLuckyGem(myMonth); // function could throw exception  
}  
catch (error) {  
  myGem = 'unknown';  
  console.error(error.message);  
}
```



# Try/Catch

```
const getNumber = async () => {  
  
  try {  
    let num = await readFileAsync('/luckyNumber.txt')  
    let success = await bookmaker.bet(num)  
  } catch (error) {  
    console.error(error.message)  
  }  
  
}  
  
getNumber()
```