Asynchronicity: concurrency. A tale of

(a.k.a: the subtle art of making a PB&J sandwich)

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/")
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo

PHP

```php
<?php

echo("Before reading file\n");
$fileContents = file_get_contents("sample.txt");
echo("After file read\n");

echo("Before HTTP request\n");
$ch = curl_init("https://swapi.co/api/planets/1/");
$data = curl_exec($ch);
curl_close($ch);
echo("After HTTP request\n");
```

Javascript

```javascript
<script type="text/javascript">

console.log("Before reading file");
getFileContent("sample.txt", () => {
  console.log("After file read");
});

echo("Before HTTP request");
const req = new XMLHttpRequest();
req.onreadystatechange = () => {
  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
    console.log("After HTTP request")
  }
};

req.open("GET", "https://swapi.co/api/planets/1/", true);
req.send(null);
</script>
```

@joel__lord
#confoo


Race Conditions

About Me

@joel__lord

joellord

Our Agenda

- Event Loop
- Callbacks
- Promises
- Generators
- Await/Async
- Events
- Reactive Events


How to make a PB&J sandwich

PB&J Sandwich

- Get the ingredients
  - Get some bread
  - Get some PB
  - Get some J
- Prepare the sandwich
  - Spread PB
  - Spread J
- Close it
- Eat it!

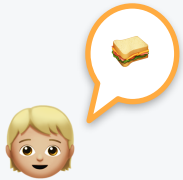
@joel__lord
#confoo


The Event Loop

Classic Architecture (PHP, Java)

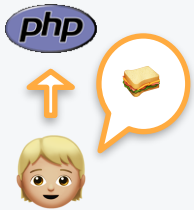
Classic Architecture (PHP, Java)

- I want a sandwich!



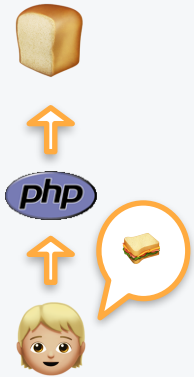
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start



Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients

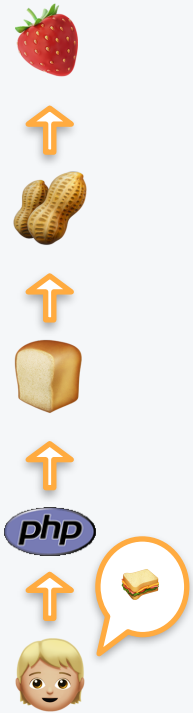


Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients

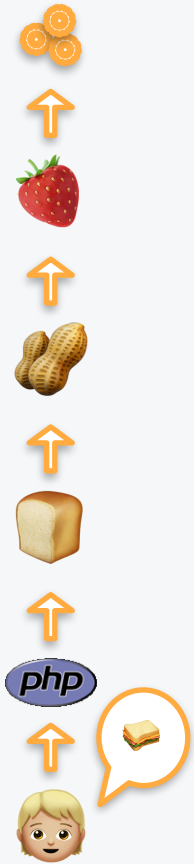
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients



Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
  - Prepares the sandwich

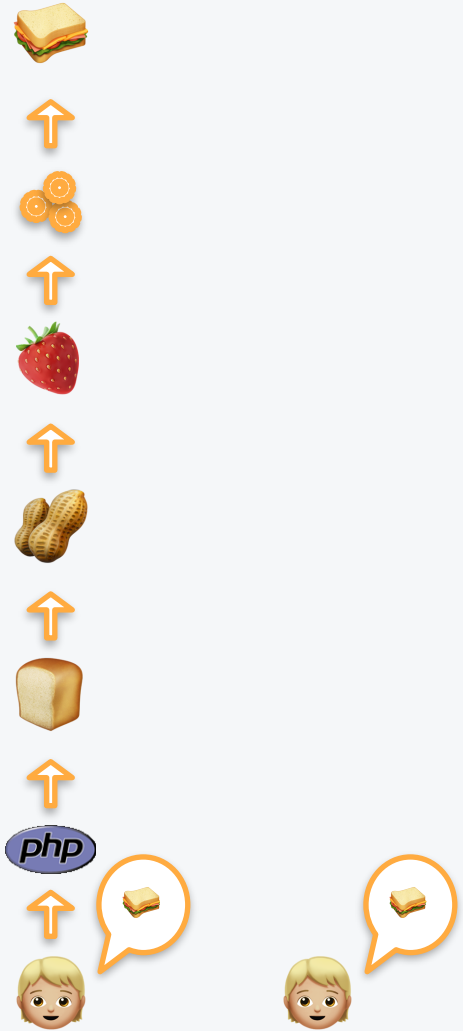


Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich

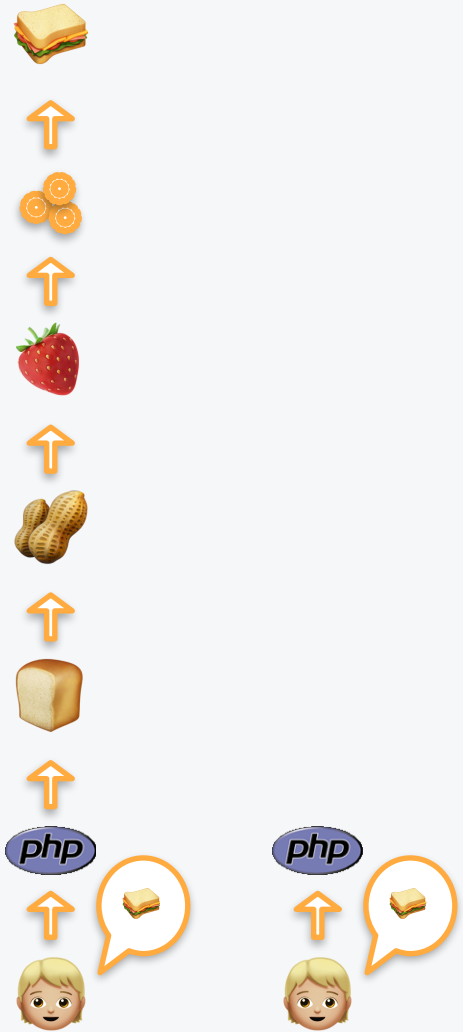
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!



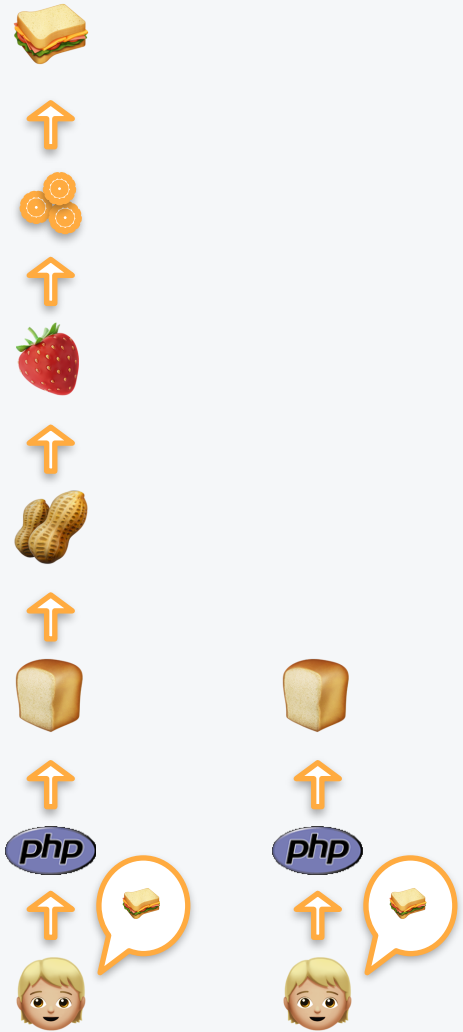
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts



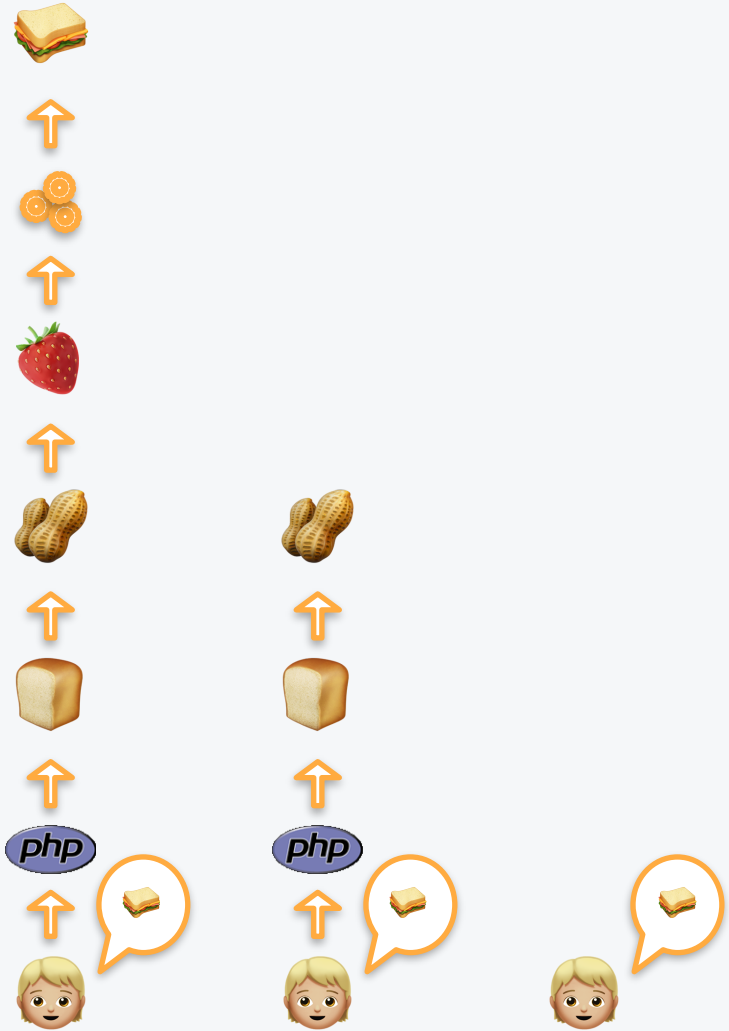
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients



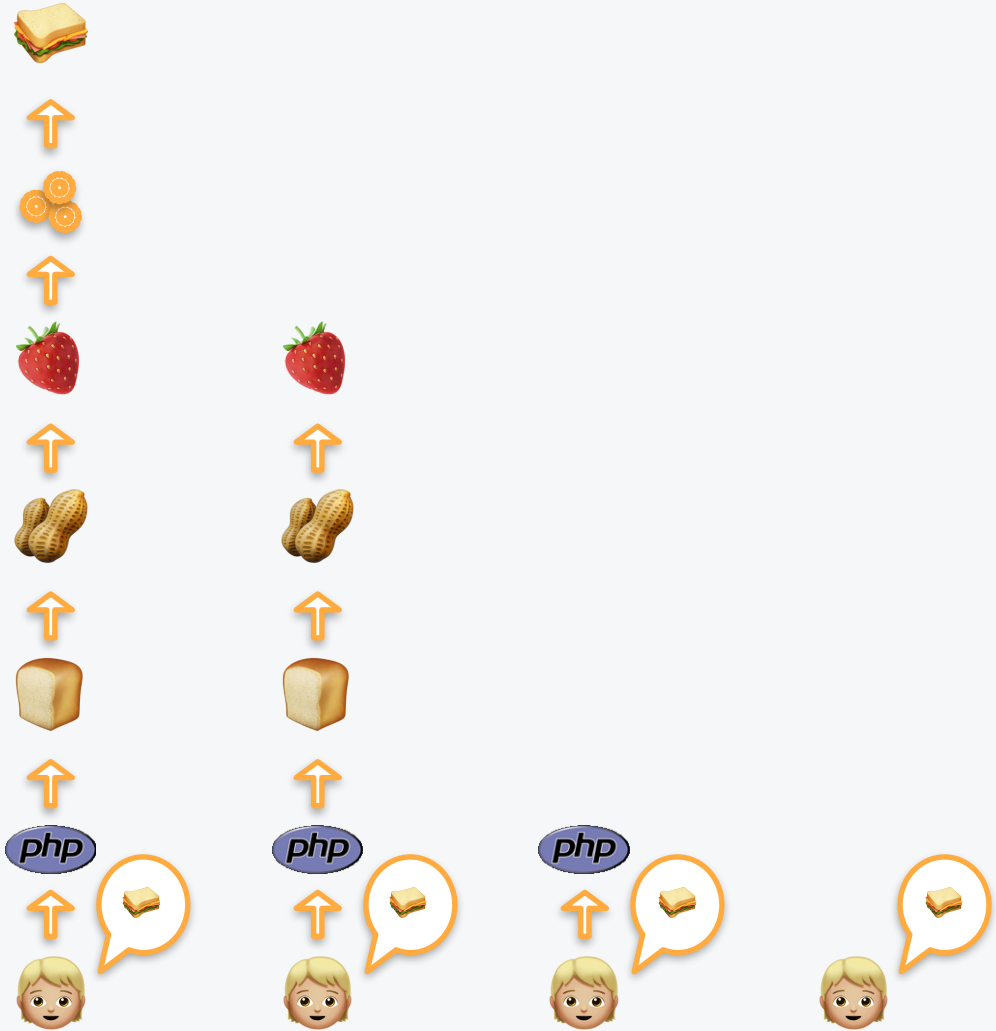
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich



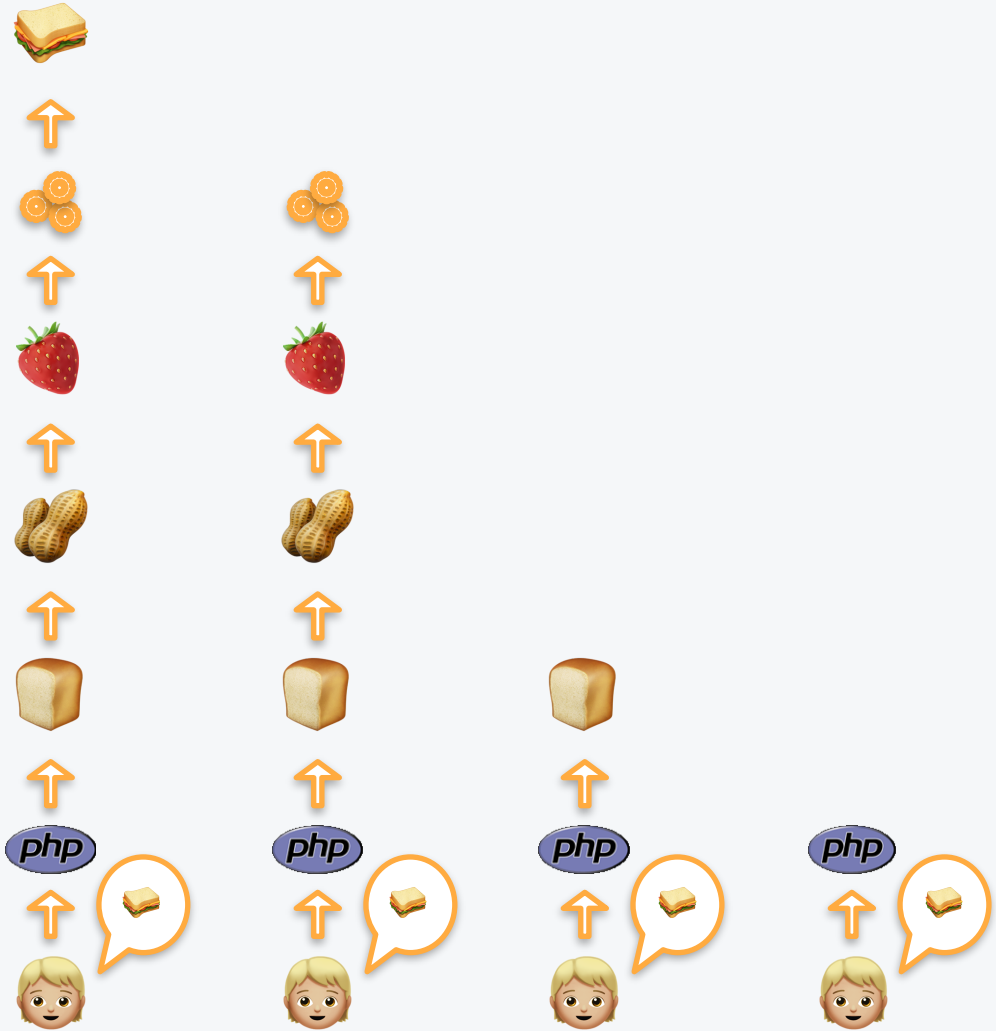
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich



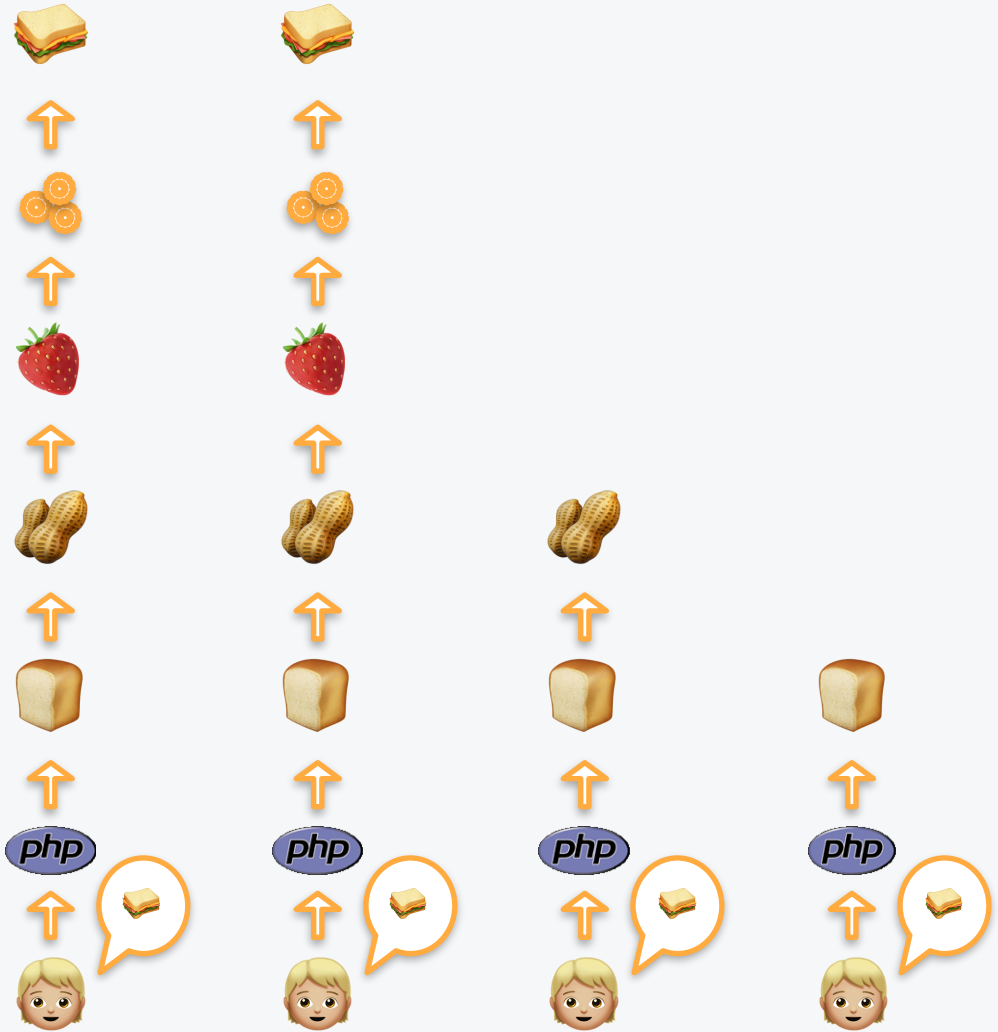
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich



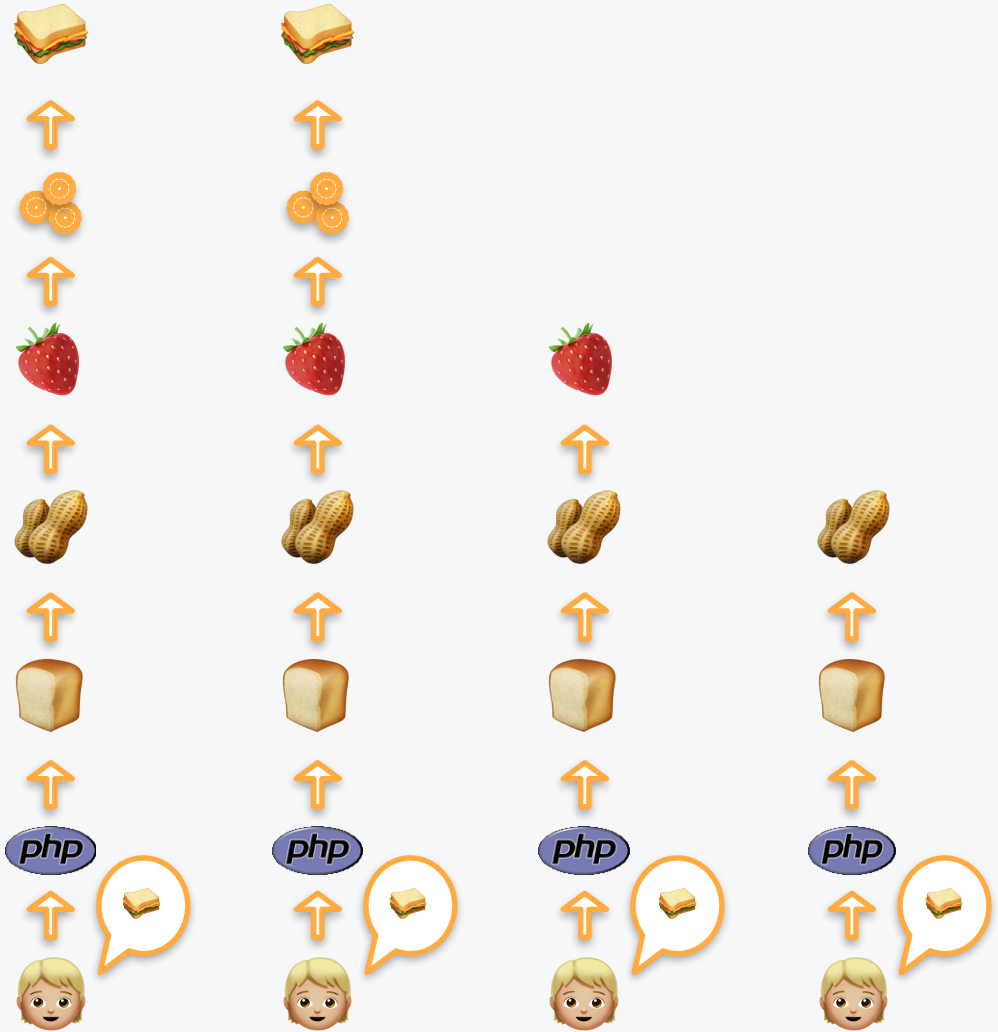
Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich



Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich



Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich

Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich

Classic Architecture (PHP, Java)

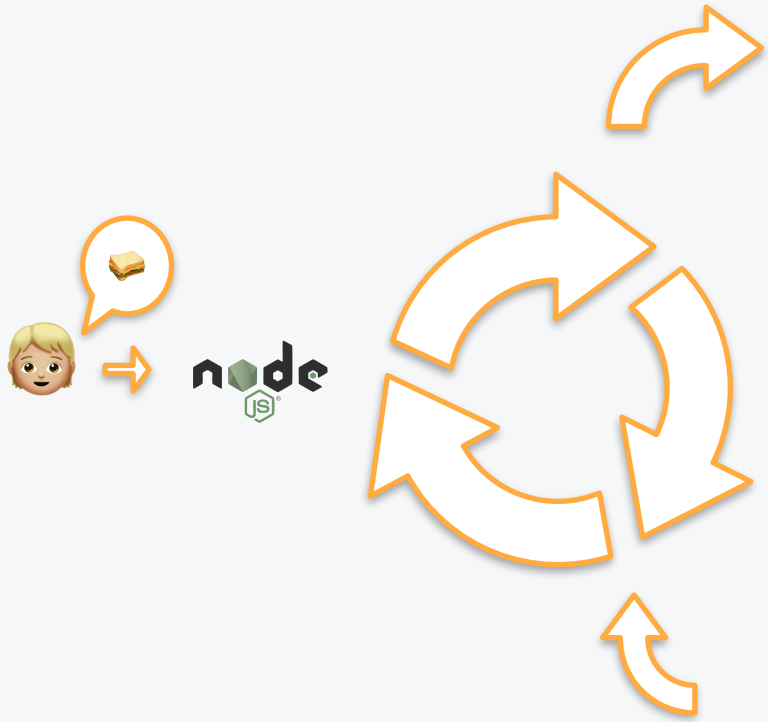
- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich

Classic Architecture (PHP, Java)

- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich

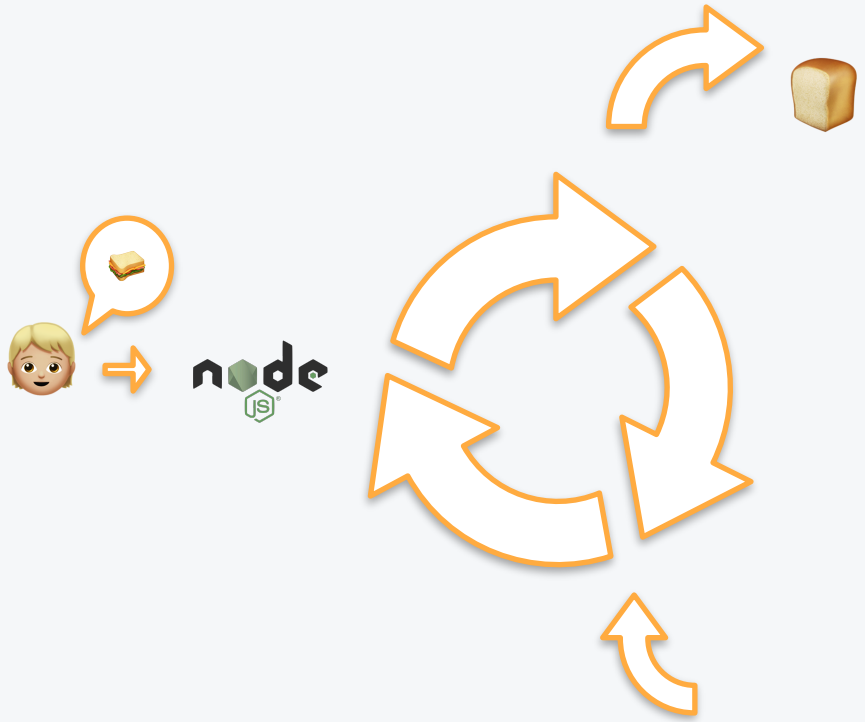
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



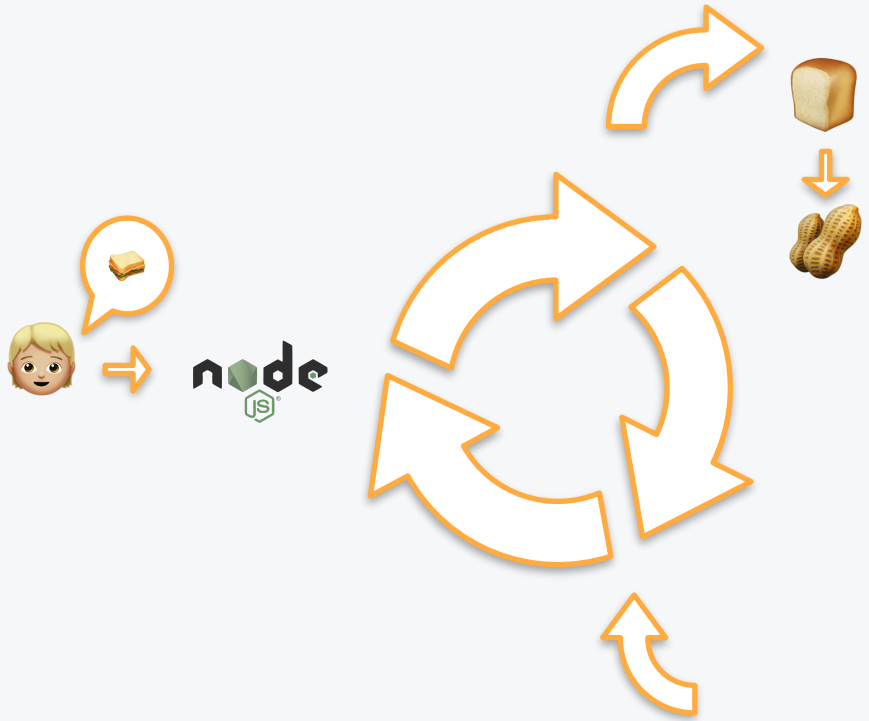
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



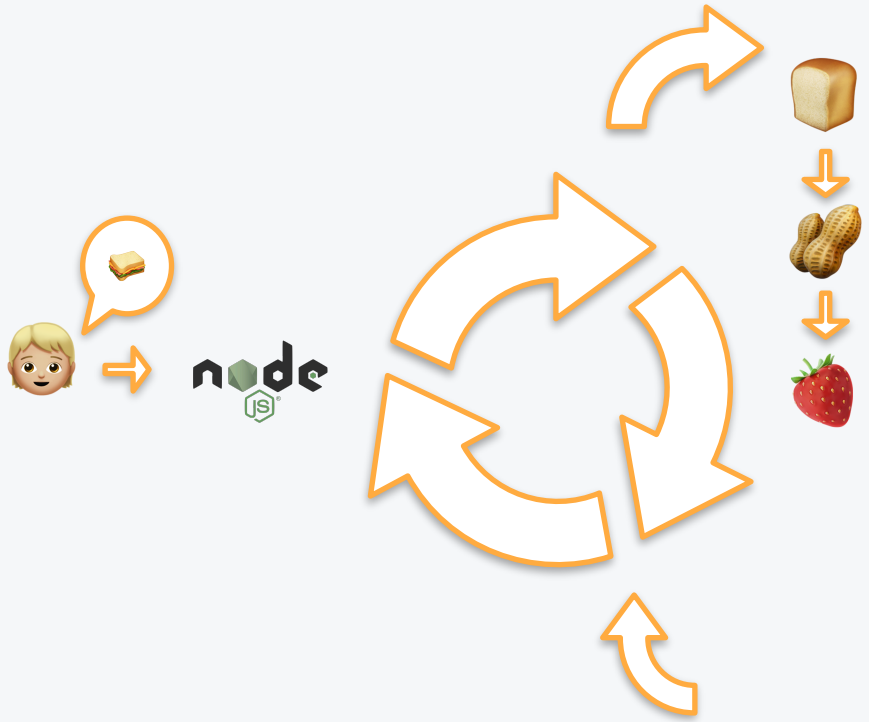
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



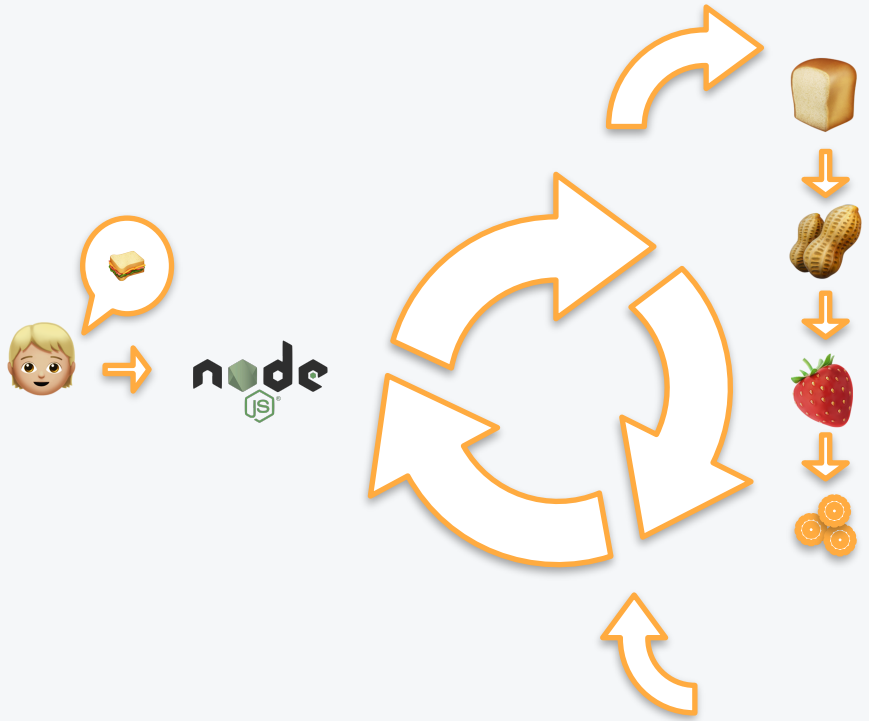
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



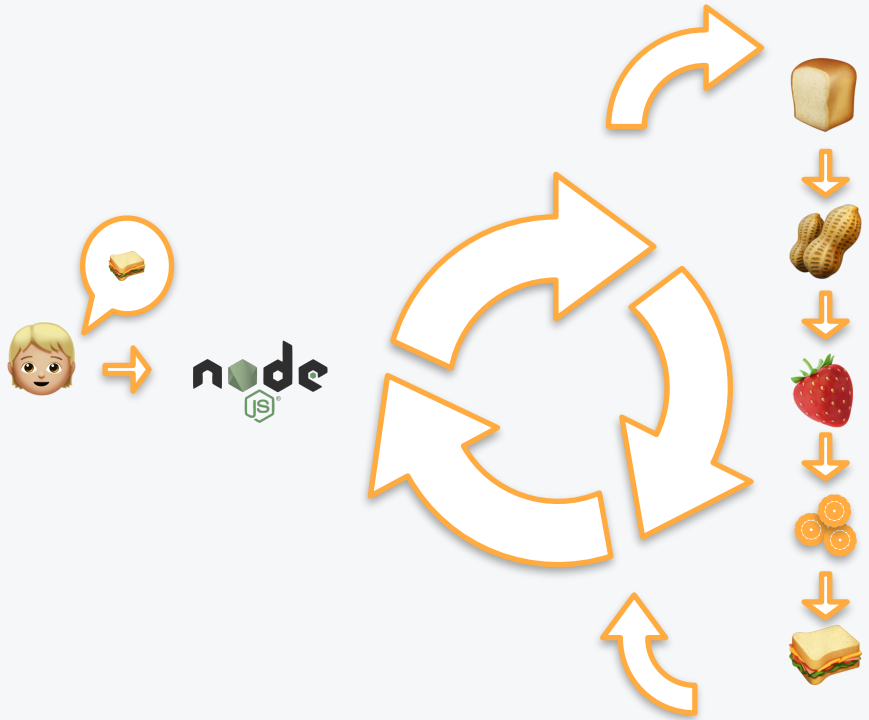
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



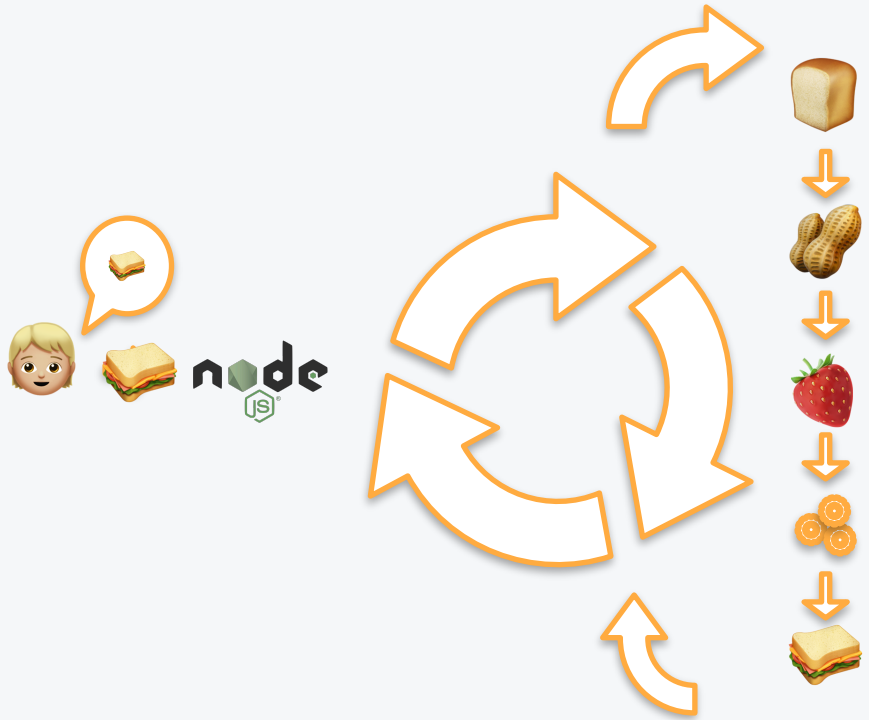
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



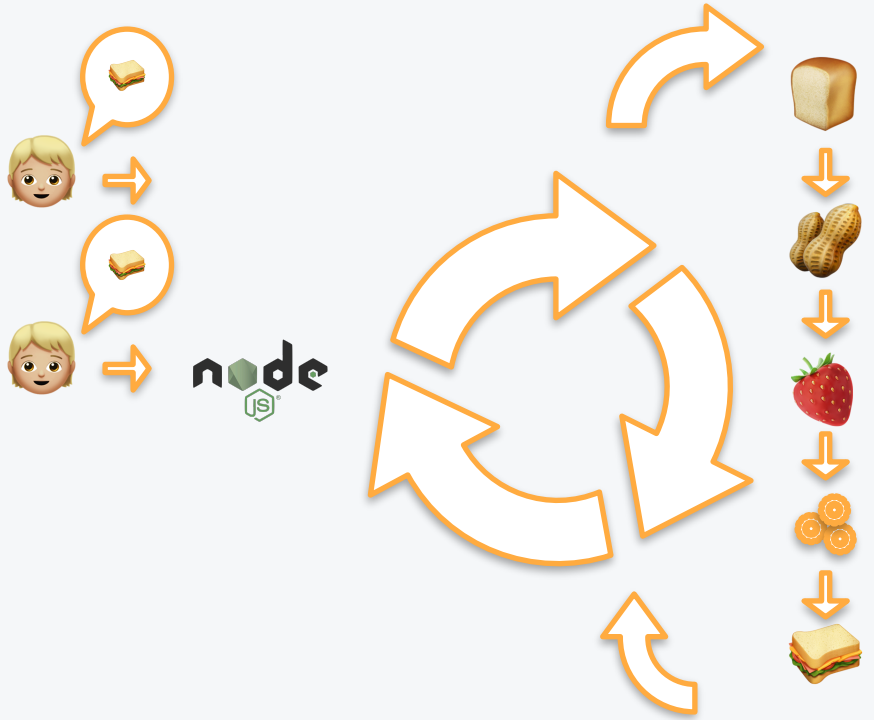
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



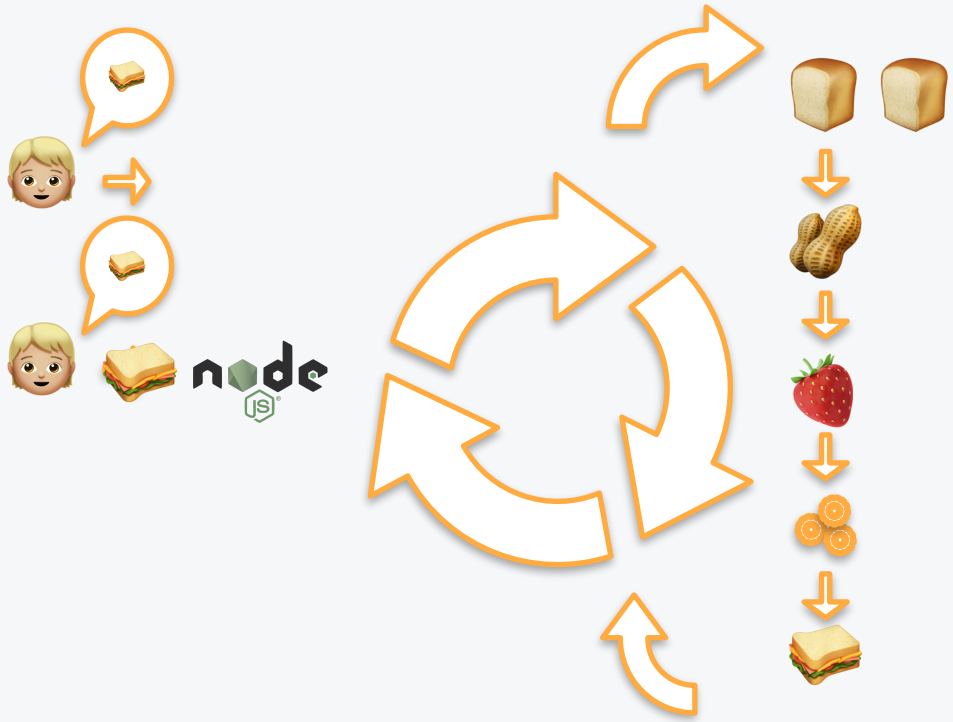
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



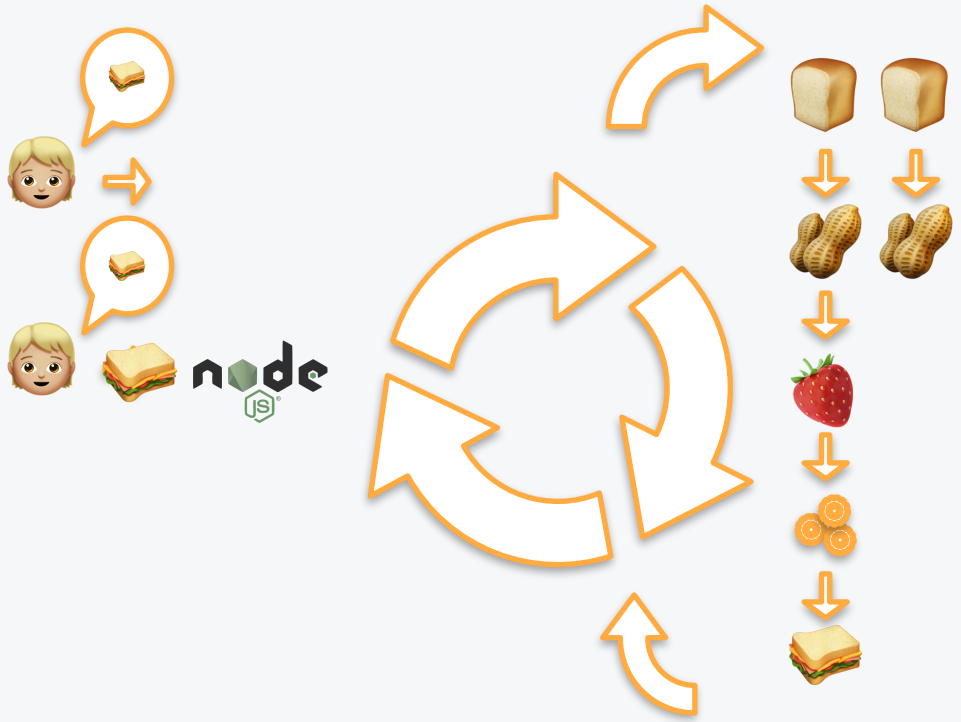
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



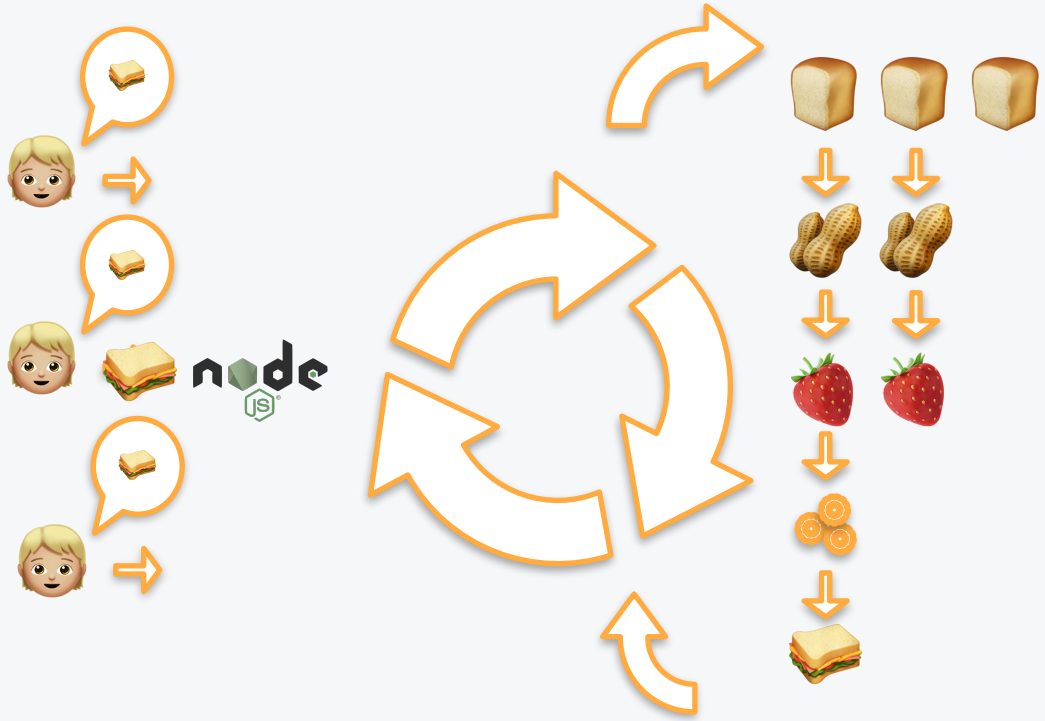
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



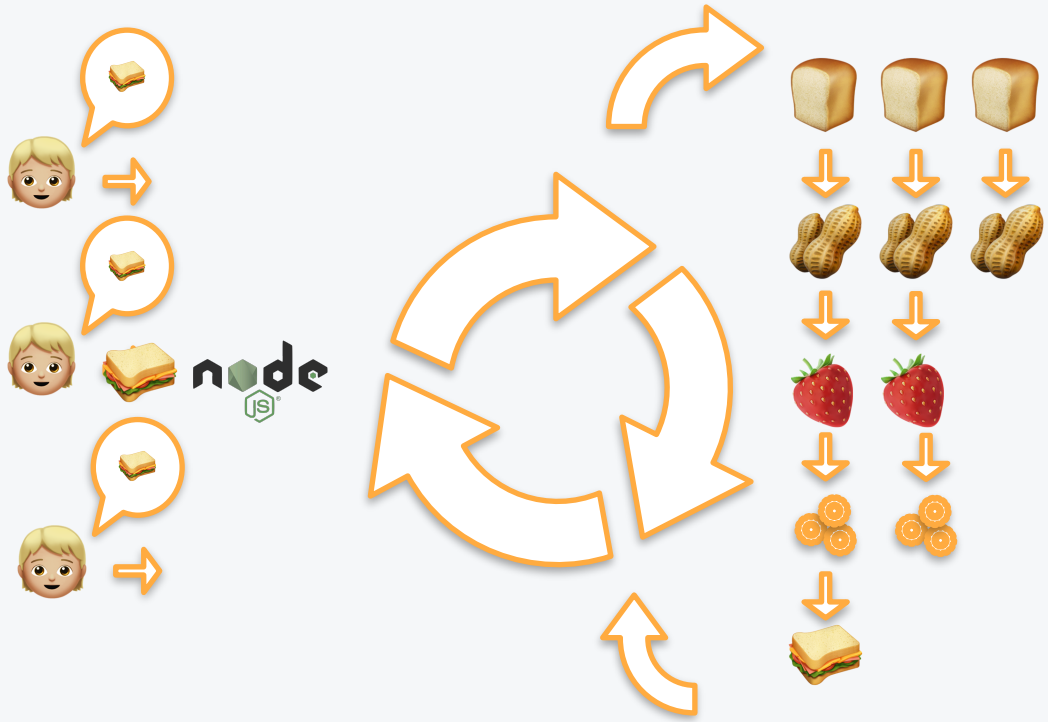
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



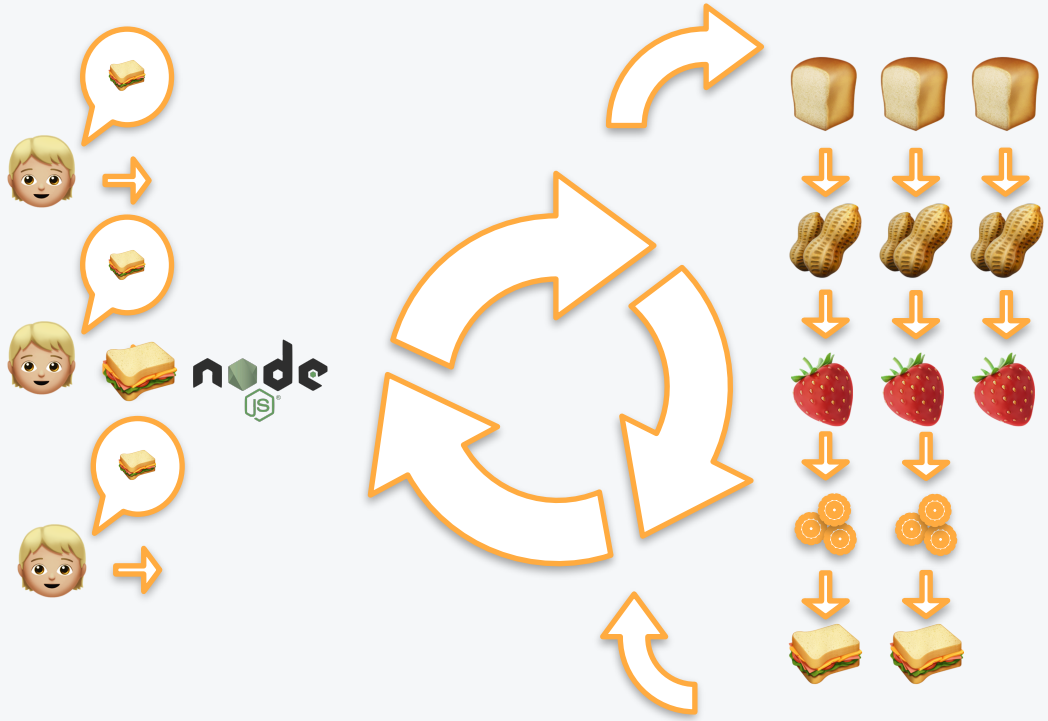
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



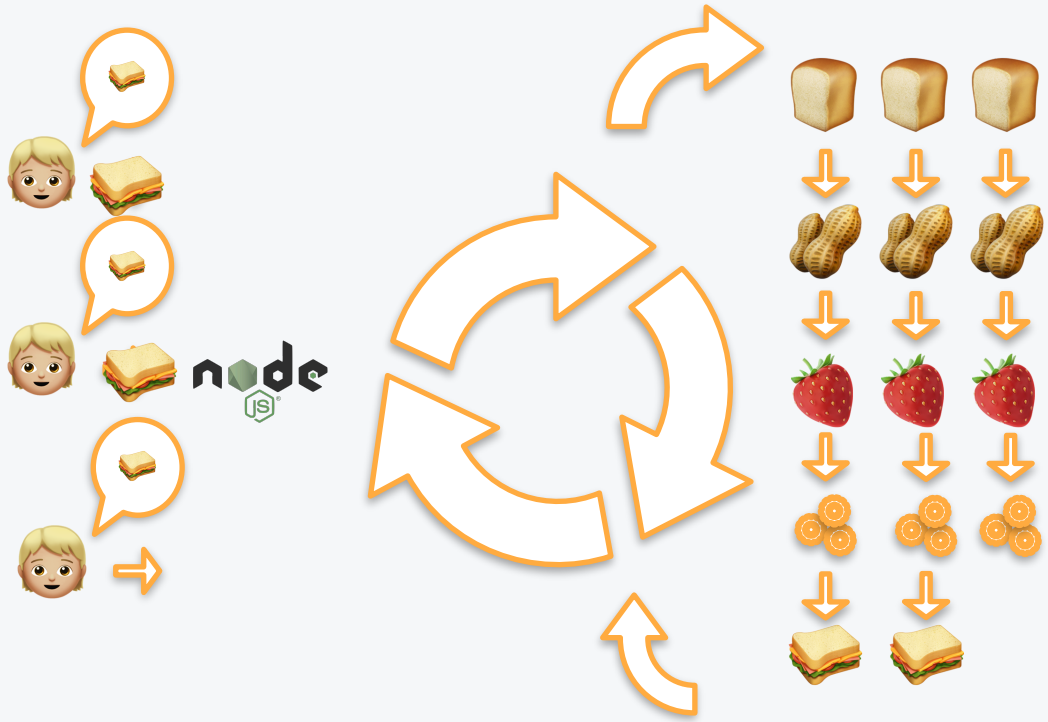
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



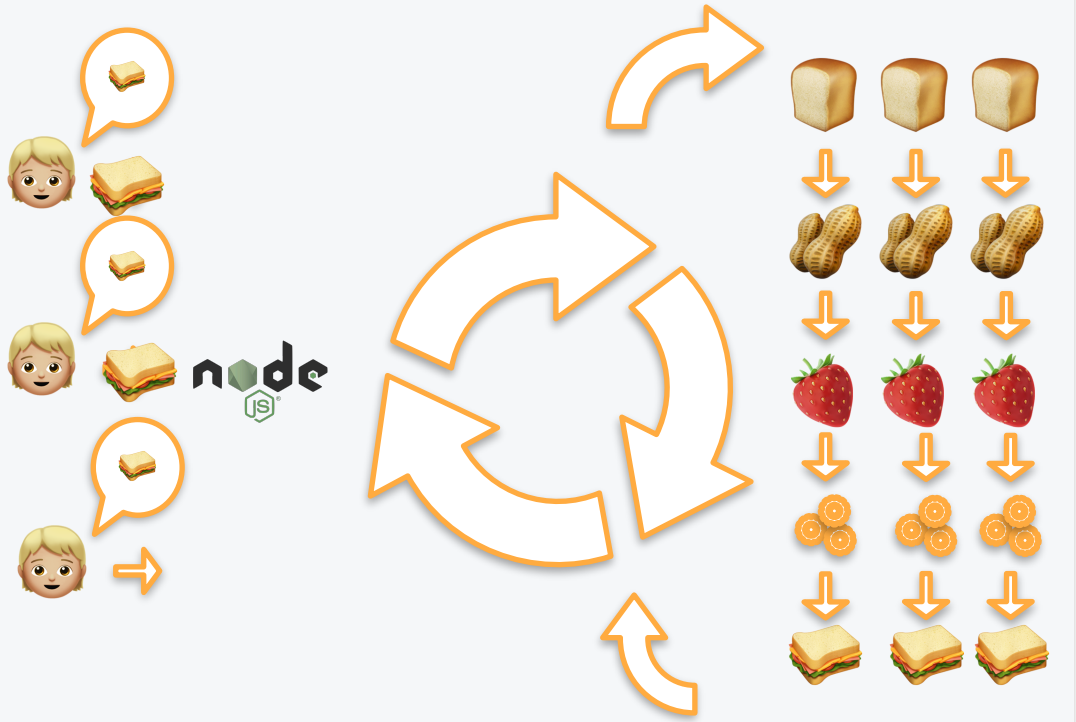
Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



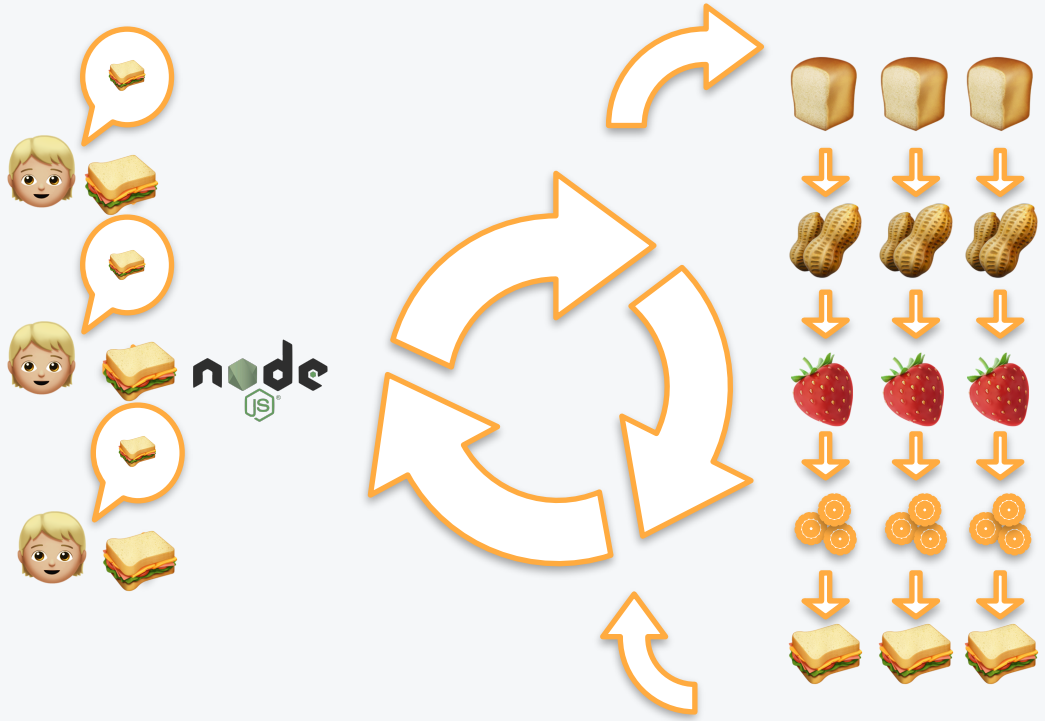
Event Loop (NodeJS)

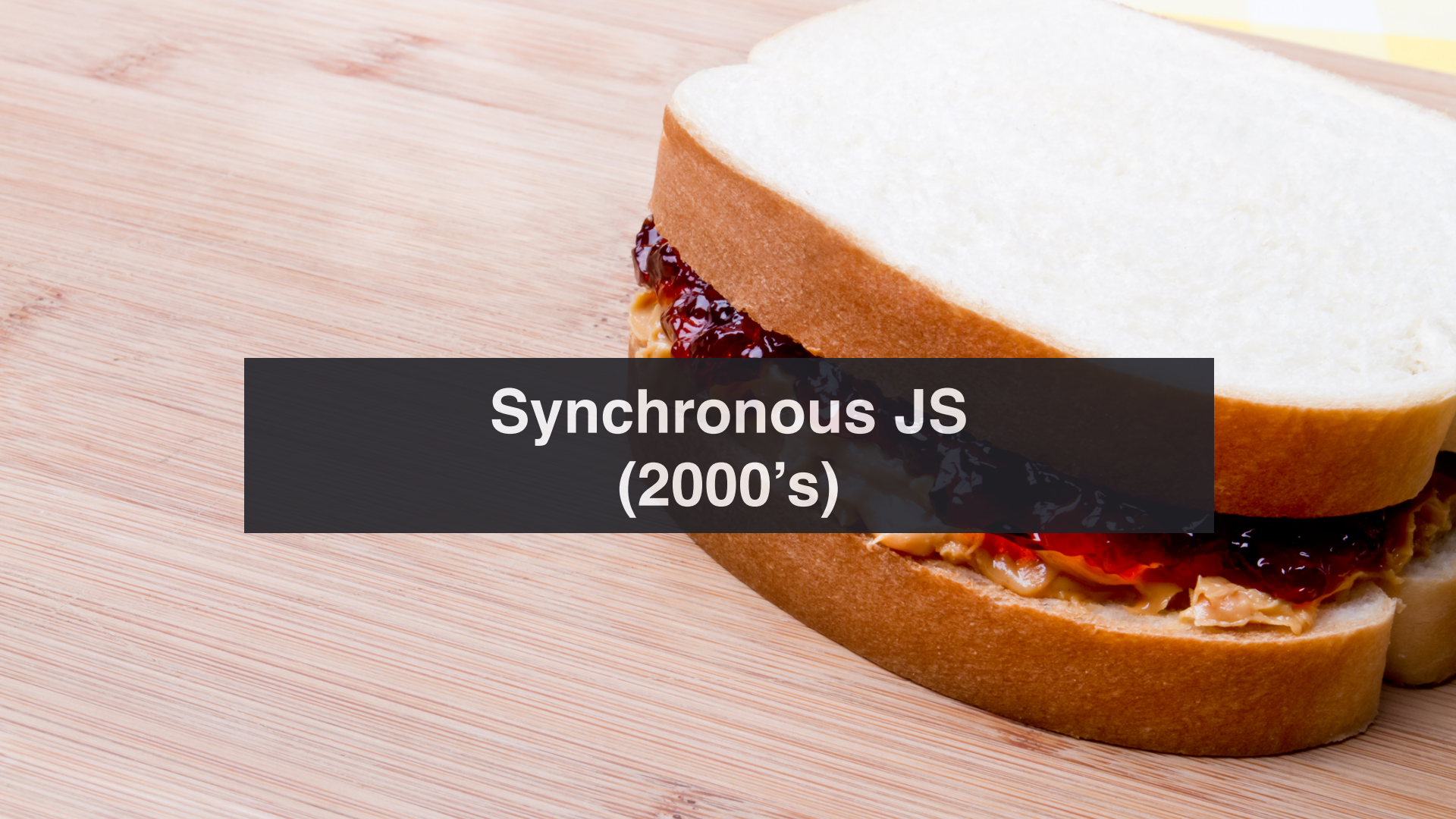
- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



Event Loop (NodeJS)

- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



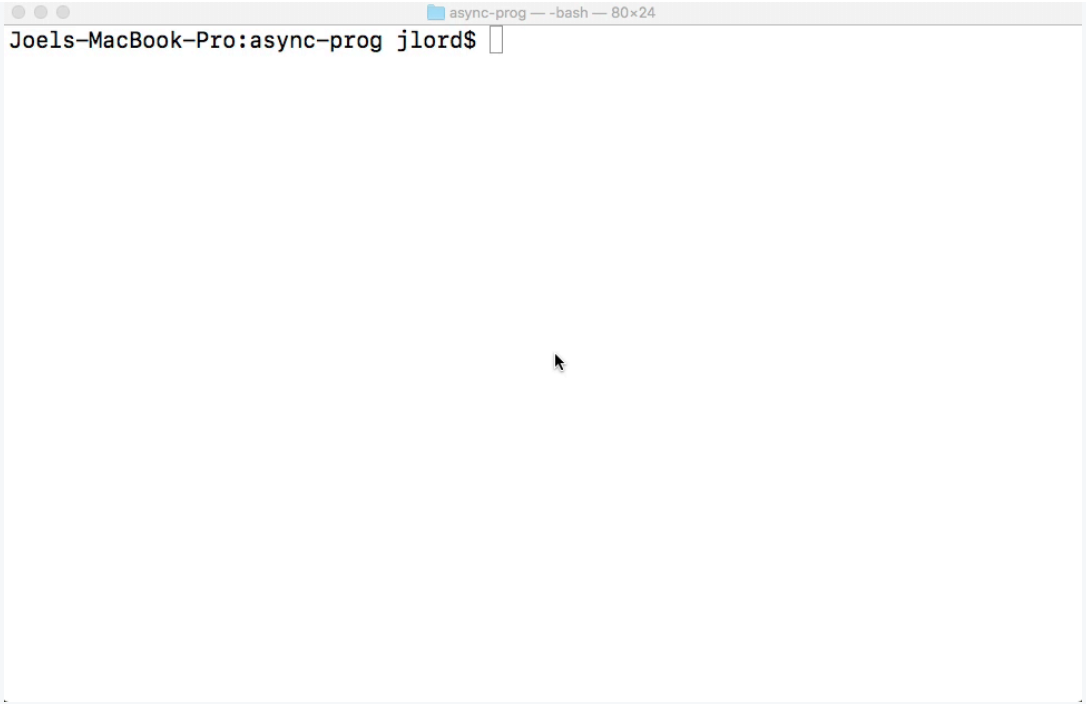

Synchronous JS
(2000’s)

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread();
PBnJMaker.fetchPeanutButter();
PBnJMaker.fetchJam();
PBnJMaker.spreadPeanutButter();
PBnJMaker.spreadJam();
let result = PBnJMaker.closeSandwich();
console.log("Eat that " + result);
```

Auth0

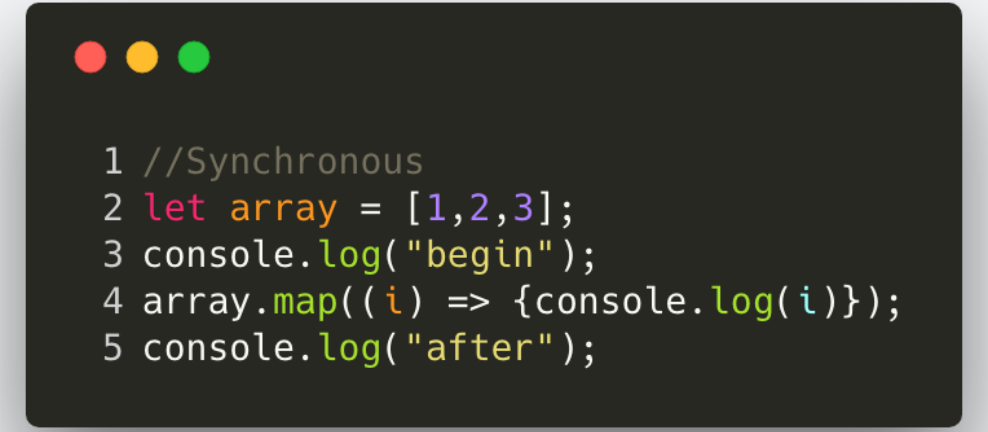
Result



Callbacks (2010's)

What is a callback

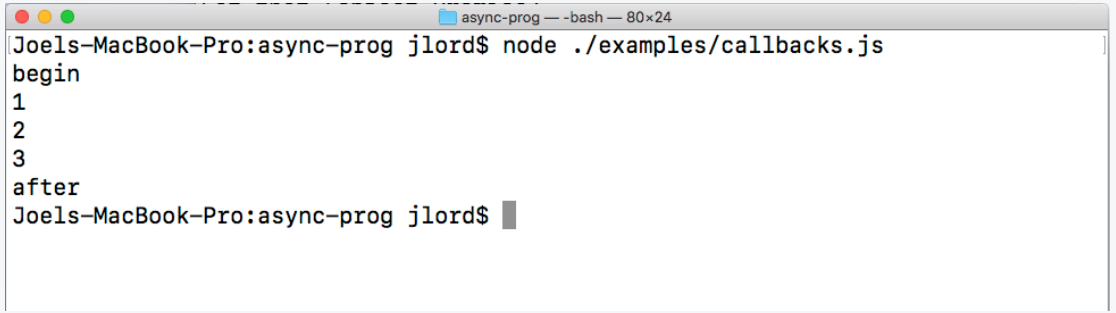
- Simply a function as an argument to a function
- Could be synchronous or asynchronous



```
1 //Synchronous
2 let array = [1,2,3];
3 console.log("begin");
4 array.map((i) => {console.log(i)});
5 console.log("after");
```

What is a callback

- Simply a function as an argument to a function
- Could be synchronous or asynchronous



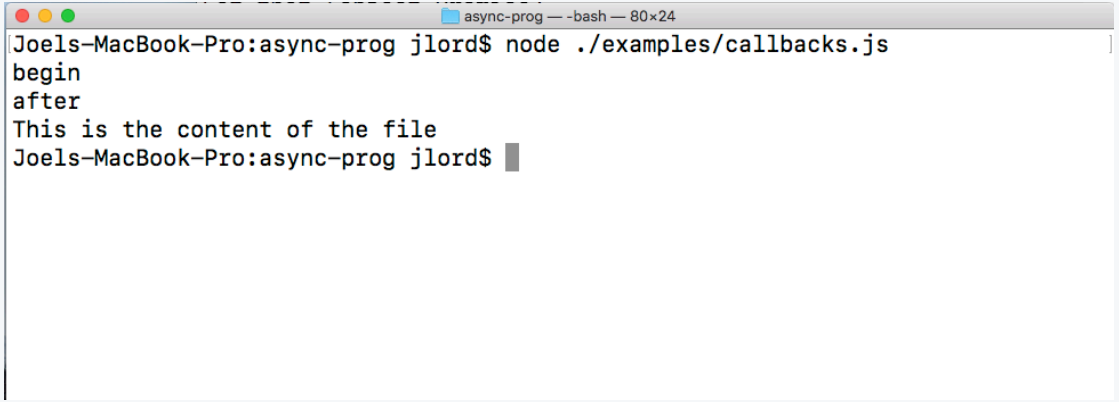
What is a callback

- Simply a function as an argument to a function
- Could be synchronous or asynchronous

```
1 //Asynchronous
2 console.log("begin");
3 fs.readFile("./cb.txt", (err, data) => {
4     console.log(data);
5 });
6 console.log("after");
```

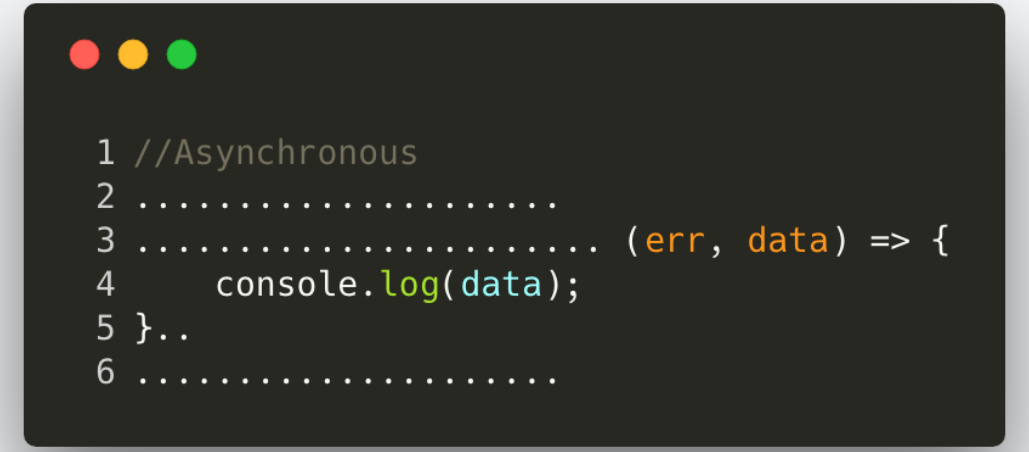
What is a callback

- Simply a function as an argument to a function
- Could be synchronous or asynchronous



What is a callback

- Simply a function as an argument to a function
- Could be synchronous or asynchronous



What is a callback

Asynchronous callback and the event loop

```
console.log("begin");

fs.readFile("./cb.txt",
(err, data) => {
    console.log(data);
});

console.log("after");
```

```
async-prog — -bash — 80×24
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
```

@joel__lord
#confoo

What is a callback

Asynchronous callback and the event loop

```
fs.readFile("./cb.txt",
(err, data) => {
    console.log(data);
});

console.log("after");
```

```
async-prog — -bash — 80×24
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
```

@joel__lord
#confoo

What is a callback

Asynchronous callback and the event loop

```
console.log("after");
```

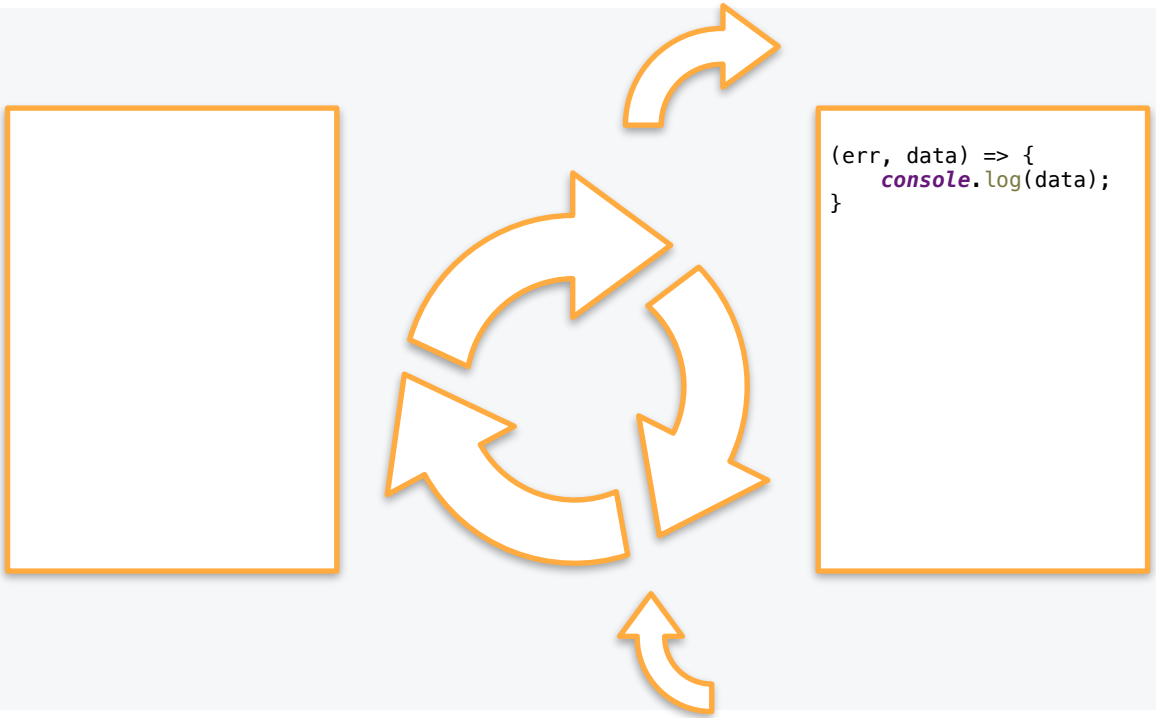
```
(err, data) => {
    console.log(data);
}
```

```
async-prog — -bash — 80×24
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
```

@joel__lord
#confoo

What is a callback

Asynchronous callback and the event loop



```
(err, data) => {
    console.log(data);
}
```

```
async-prog — -bash — 80×24
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
```

What is a callback

Asynchronous callback and the event loop

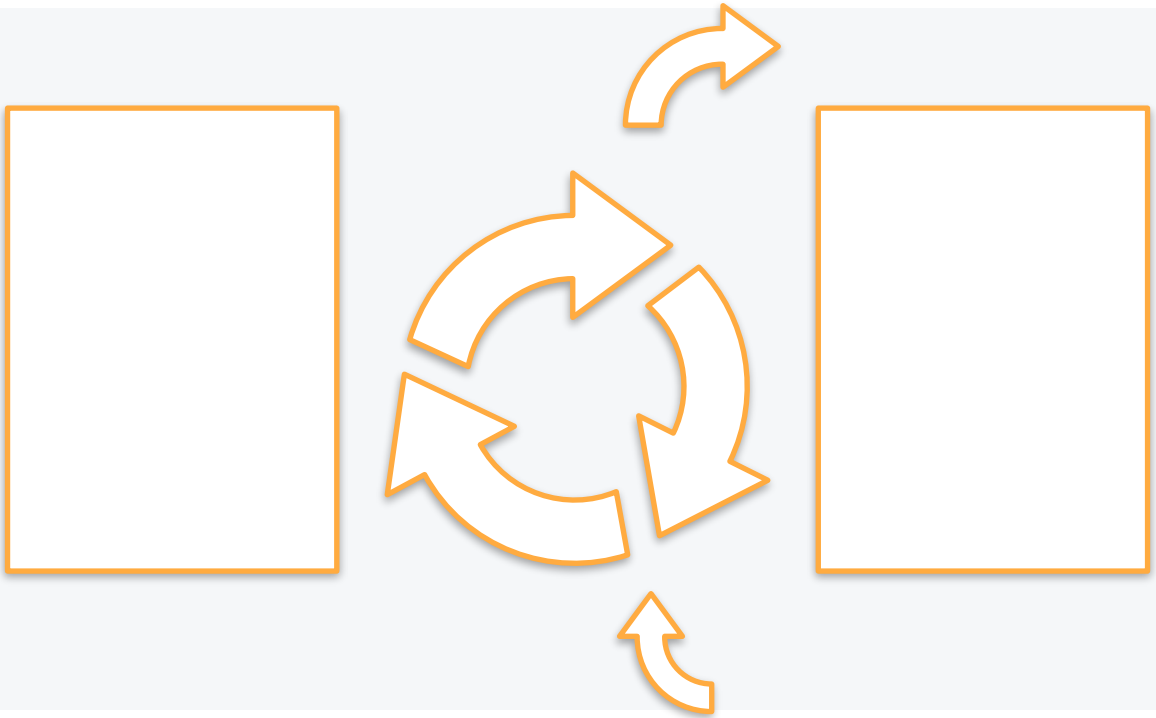
```
(err, data) => {
    console.log(data);
}
```

```
async-prog — -bash — 80×24
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
after
```

@joel__lord
#confoo

What is a callback

Asynchronous callback and the event loop



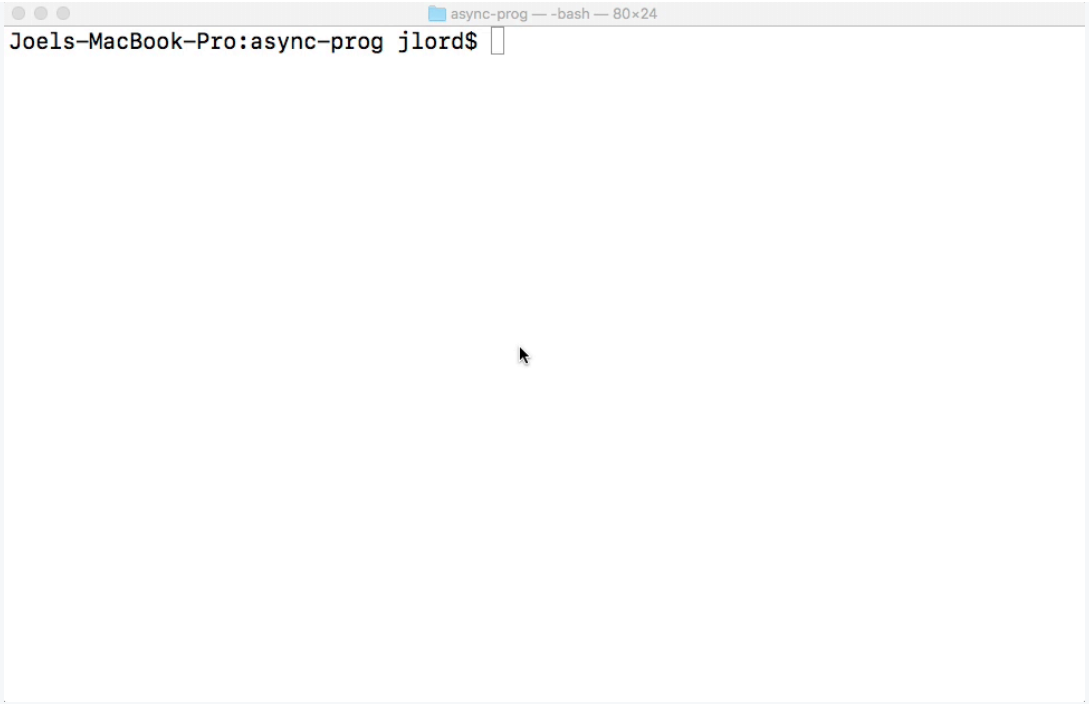
```
async-prog — -bash — 80×24
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
after
This is the content of the file
Joels-MacBook-Pro:async-prog jlord$
```

Code

```
let PBnJMaker = require("../pbnjmaker");

PBnJMaker.fetchBread(() => {
    PBnJMaker.fetchPeanutButter(() => {
        PBnJMaker.fetchJam(() => {
            PBnJMaker.spreadPeanutButter(() => {
                PBnJMaker.spreadJam(() => {
                    PBnJMaker.closeSandwich((err, output) => {
                        console.log("Eat that " + output);
                    });
                });
            });
        });
    });
});
```

Auth0

Result



Callbacks

- It works!
- but…

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.simulateError();

PBnJMaker.fetchBread((err) => {
    if (err) {
        console.log("An error occurred while fetching the ingredients");
    } else {
        PBnJMaker.fetchPeanutButter((err) => {
            if (err) {
                console.log("An error occurred while fetching the ingredients");
            } else {
                PBnJMaker.fetchJam((err) => {
                    if (err) {
                        console.log("An error occurred while fetching the ingredients");
                    } else {
                        PBnJMaker.spreadPeanutButter((err) => {
                            if (err) {
                                console.log("An error occurred while preparing the sandwich");
                            } else {
                                PBnJMaker.spreadJam((err) => {
                                    if (err) {
                                        console.log("An error occurred while preparing the sandwich");
                                    } else {
                                        PBnJMaker.closeSandwich((err, result) => {
                                            if (err) {
                                                console.log("An error occurred");
                                            } else {
                                                console.log("Eat that " + result);
                                            }
                                        });
                                    }
                                });
                            }
                        });
                    }
                });
            }
        });
    }
});
```

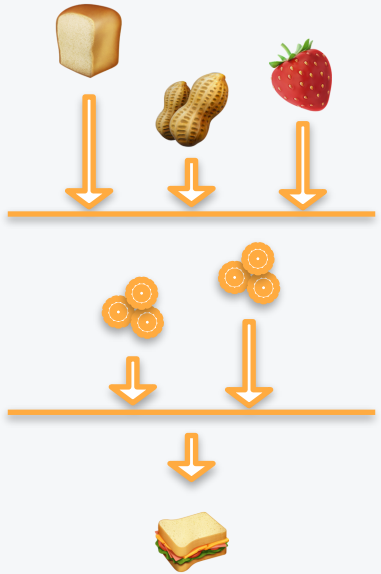
Callback Hell

```
let PBnJMaker = require("../pbnjmaker");

PBnJMaker.simulateError();

PBnJMaker.fetchBread((err) => {
    if (err) {
        console.log("An error occurred while fetching the ingredients");
    } else {
        PBnJMaker.fetchPeanutButter((err) => {
            if (err) {
                console.log("An error occurred while fetching the ingredients");
            } else {
                PBnJMaker.fetchJam((err) => {
                    if (err) {
                        console.log("An error occurred while fetching the ingredients");
                    } else {
                        PBnJMaker.spreadPeanutButter((err) => {
                            if (err) {
                                console.log("An error occurred while preparing the sandwich");
                            } else {
                                PBnJMaker.spreadJam((err) => {
```


Parallel events

Step by step

Parallel events

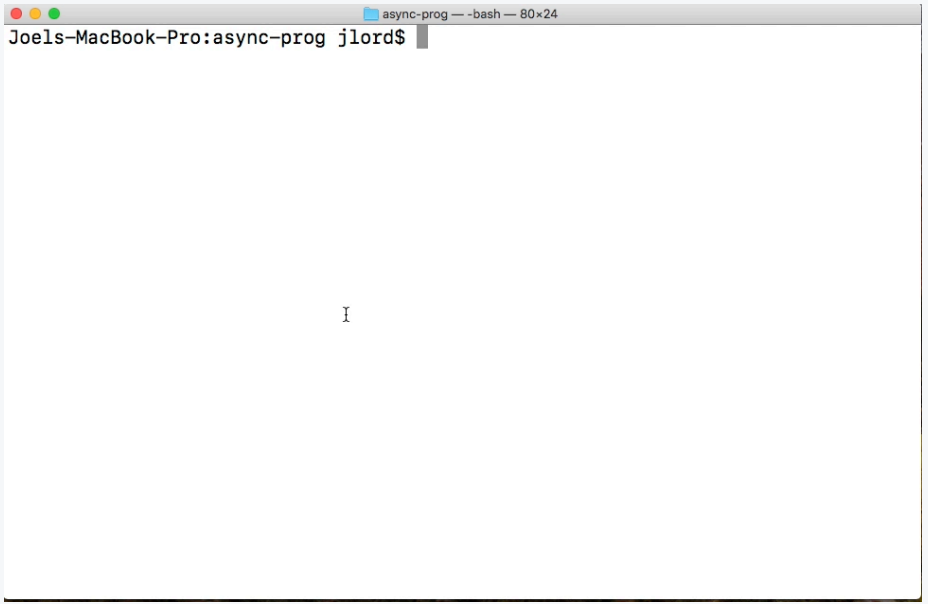


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result

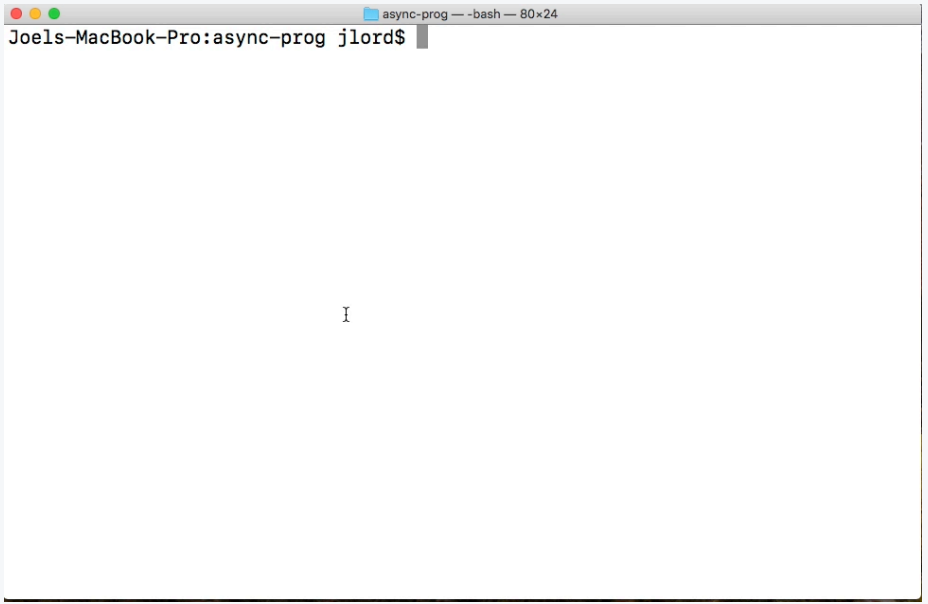


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result

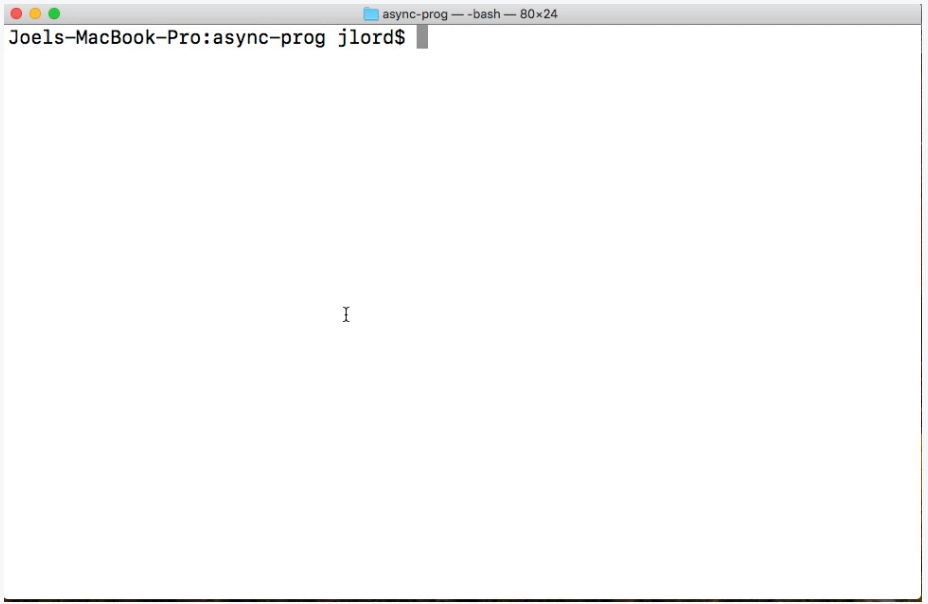


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result

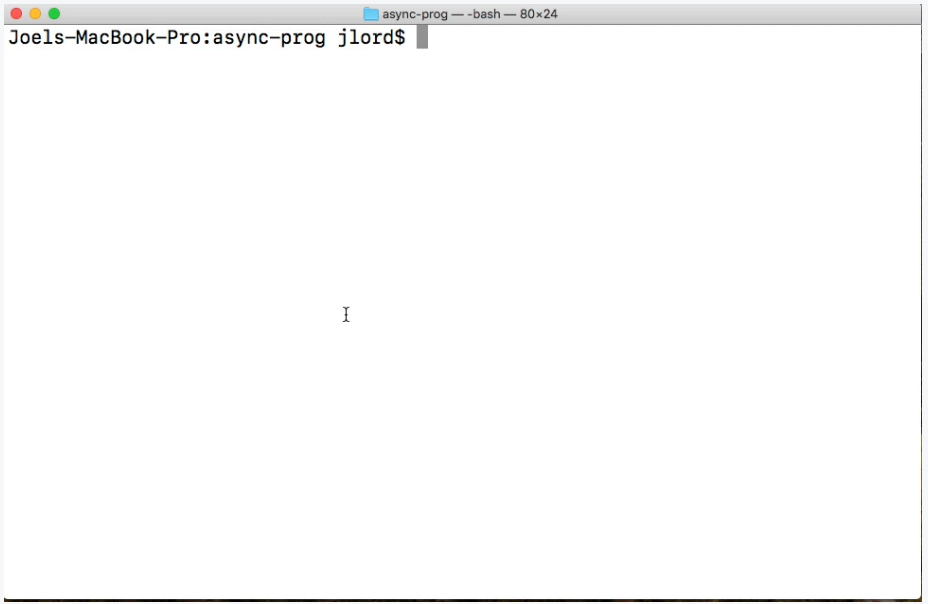


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result

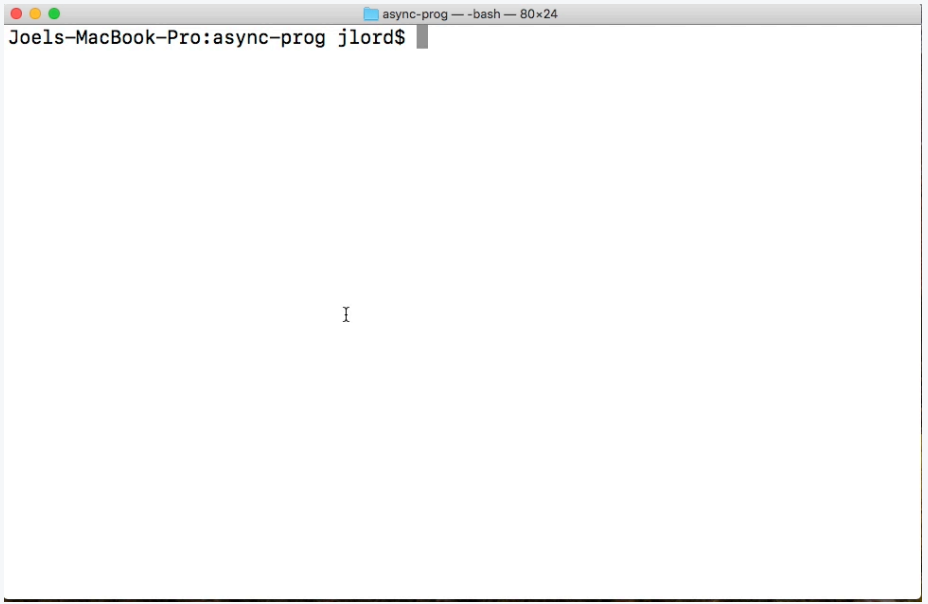


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result

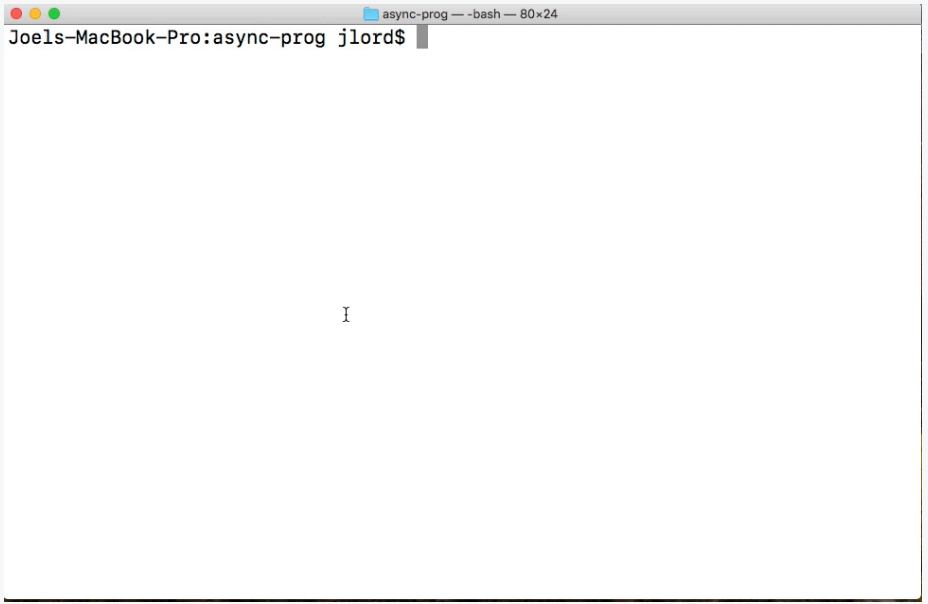


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result

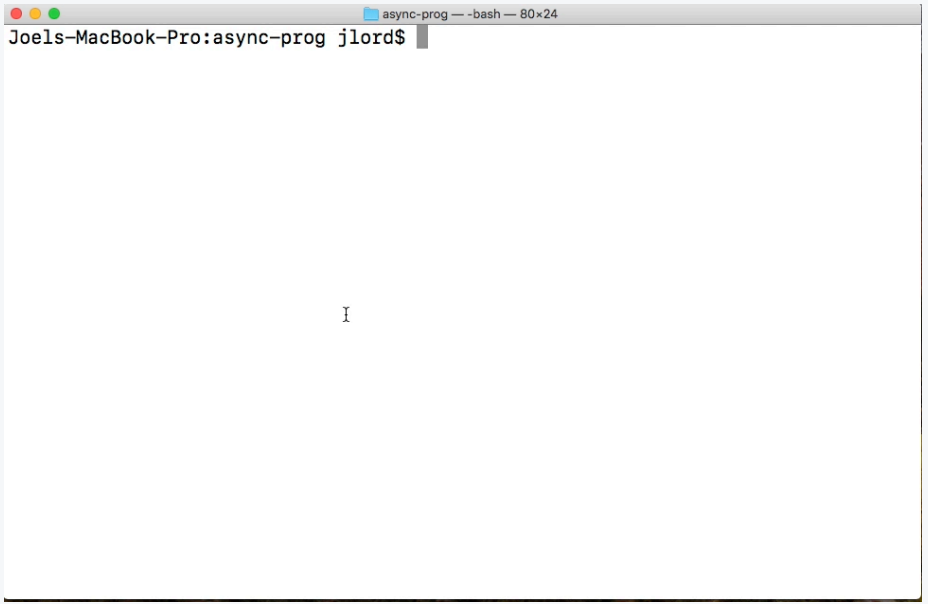


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result

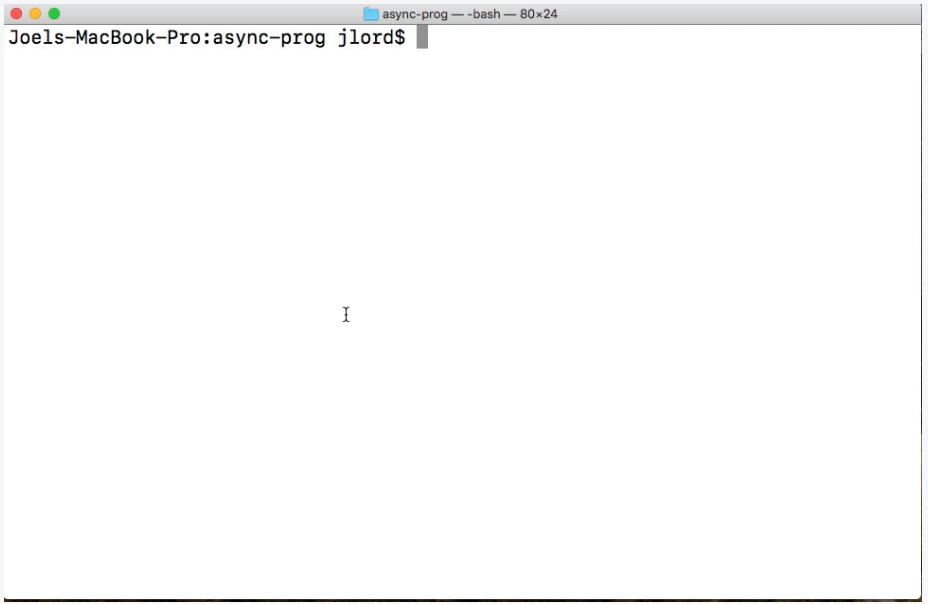


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result

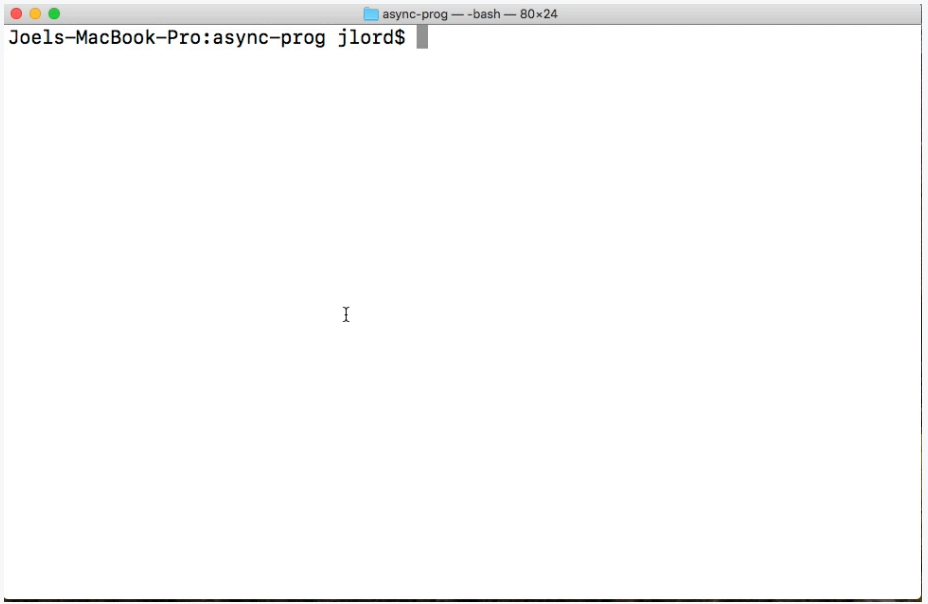


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let ingredients = 0;
let steps = 0;
// PBnJMaker.simulateError();

function fetchIngredients() {
    PBnJMaker.fetchBread(handleIngredients);
    PBnJMaker.fetchPeanutButter(handleIngredients);
    PBnJMaker.fetchJam(handleIngredients);
}

function handleIngredients(err) {
    if (err) return console.log("An error occurred while fetching the ingredients");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}

function prepareSandwich() {
    PBnJMaker.spreadPeanutButter(preparationStepCompleted);
    PBnJMaker.spreadJam(preparationStepCompleted);
}

function preparationStepCompleted(err) {
    if (err) return console.log("An error occurred while preparing the sandwich");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}

function completeSandwich(err) {
    PBnJMaker.closeSandwich((err, result) => {
        if (err) {
            console.log("An error occurred");
        } else {
            console.log("Eat that " + result);
        }
    });
}

fetchIngredients();
```

Result



@joel__lord
#confoo

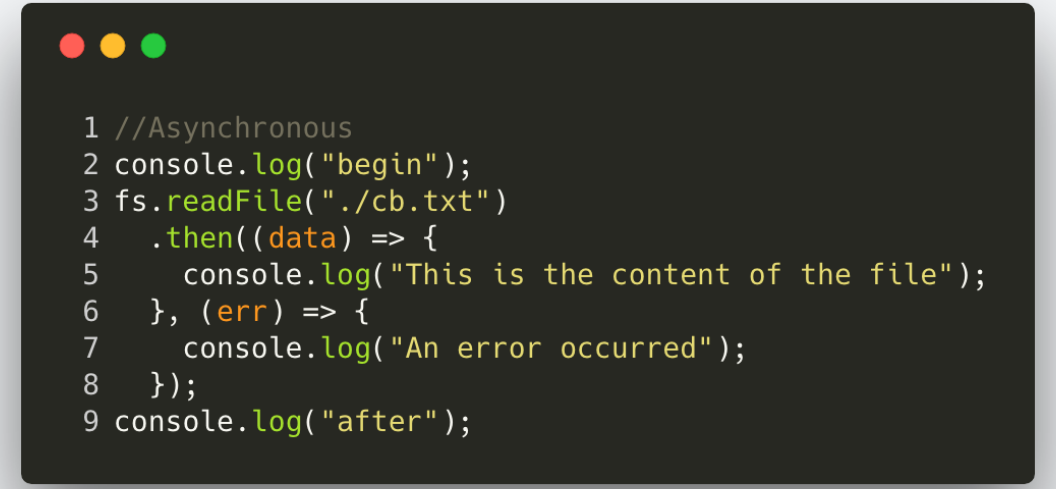

Promises
(2014)

What is a promise?

- A representation of a callback
- Returns a .then() method
- Error handling simplified

What is a promise?

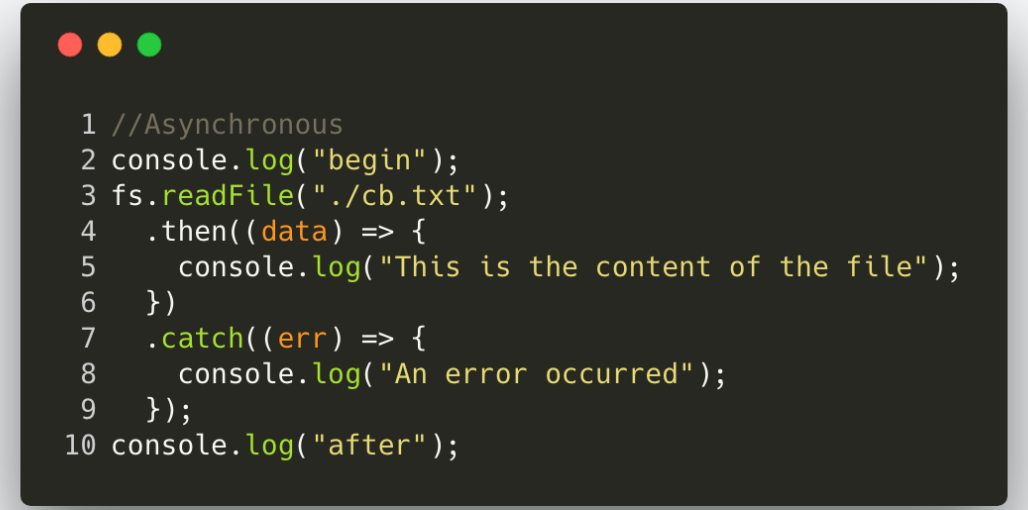
- A representation of a callback
- Returns a .then() method
- Error handling simplified



```
1 //Asynchronous
2 console.log("begin");
3 fs.readFile("./cb.txt")
4   .then((data) => {
5     console.log("This is the content of the file");
6   }, (err) => {
7     console.log("An error occurred");
8   });
9 console.log("after");
```

What is a promise?

- A representation of a callback
- Returns a .then() method
- Error handling simplified

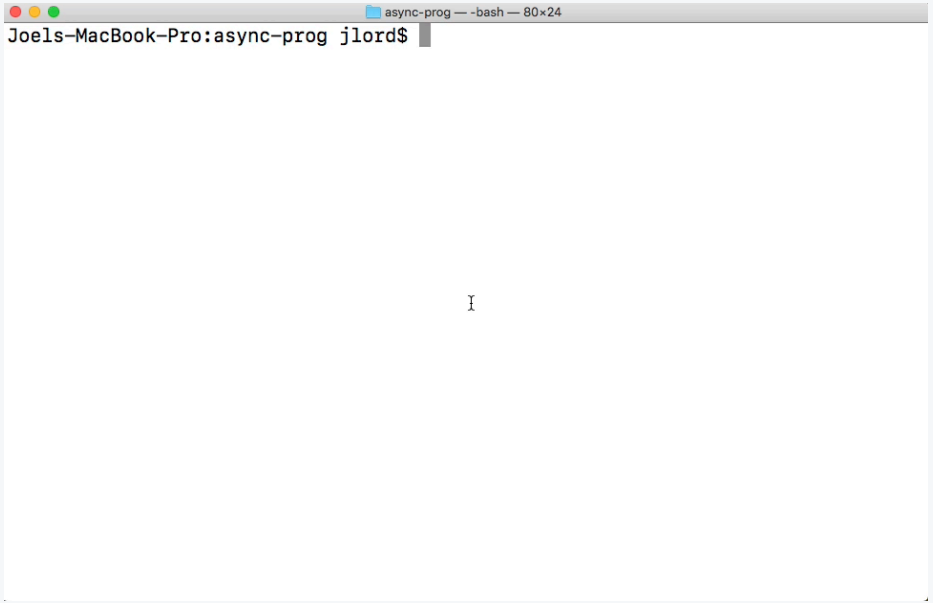


```
1 //Asynchronous
2 console.log("begin");
3 fs.readFile("./cb.txt");
4   .then((data) => {
5     console.log("This is the content of the file");
6   })
7   .catch((err) => {
8     console.log("An error occurred");
9   });
10 console.log("after");
```

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

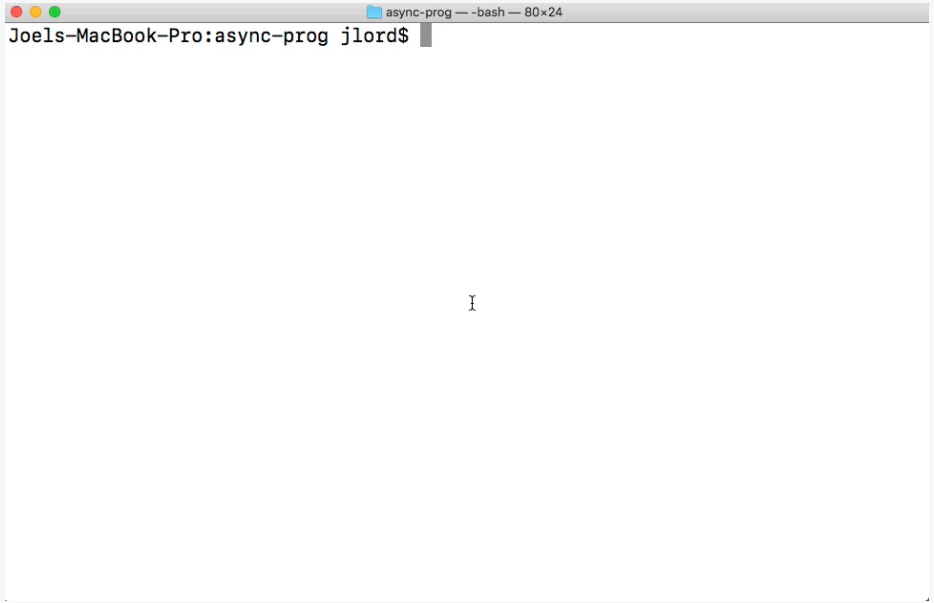
Result



Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

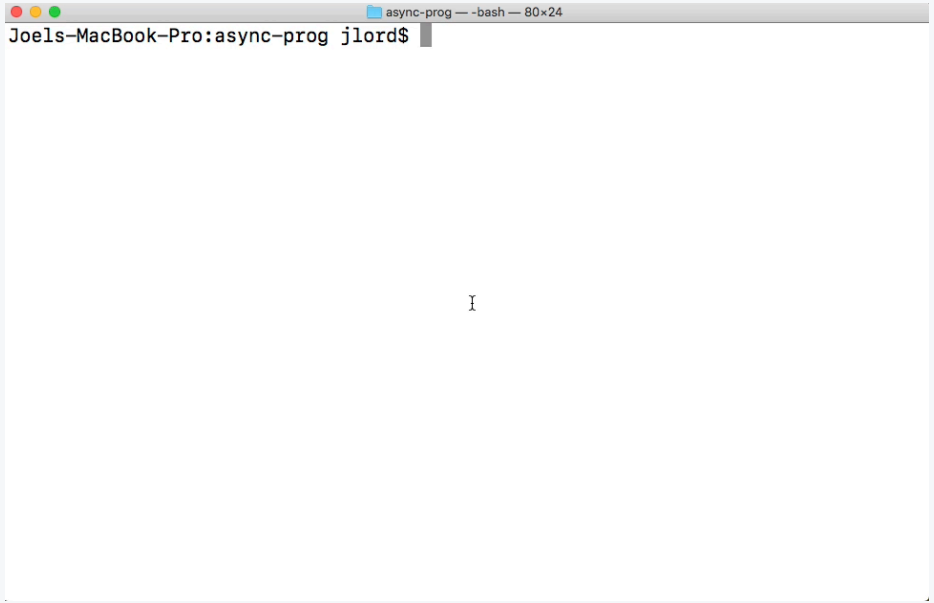
Result



Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

Result

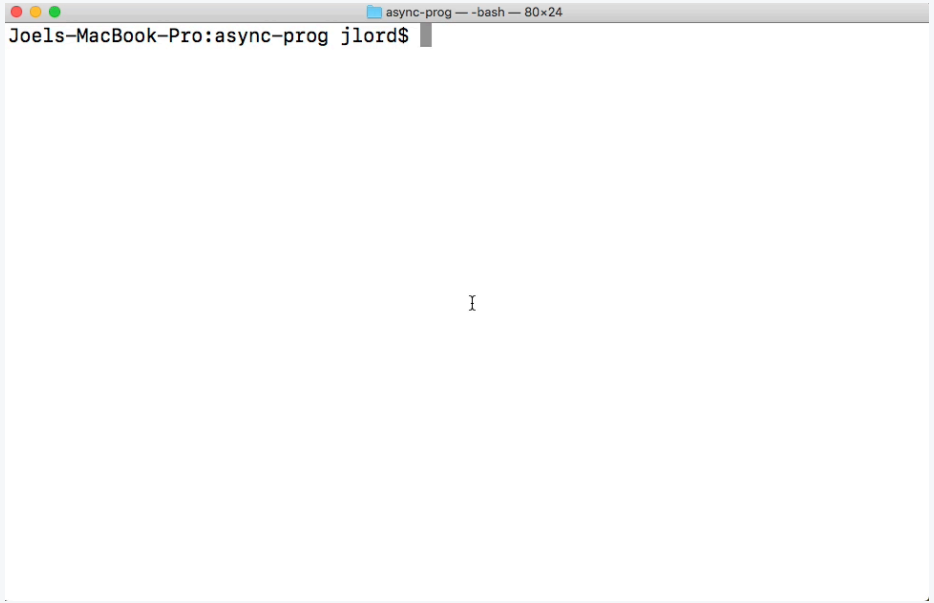


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

Result

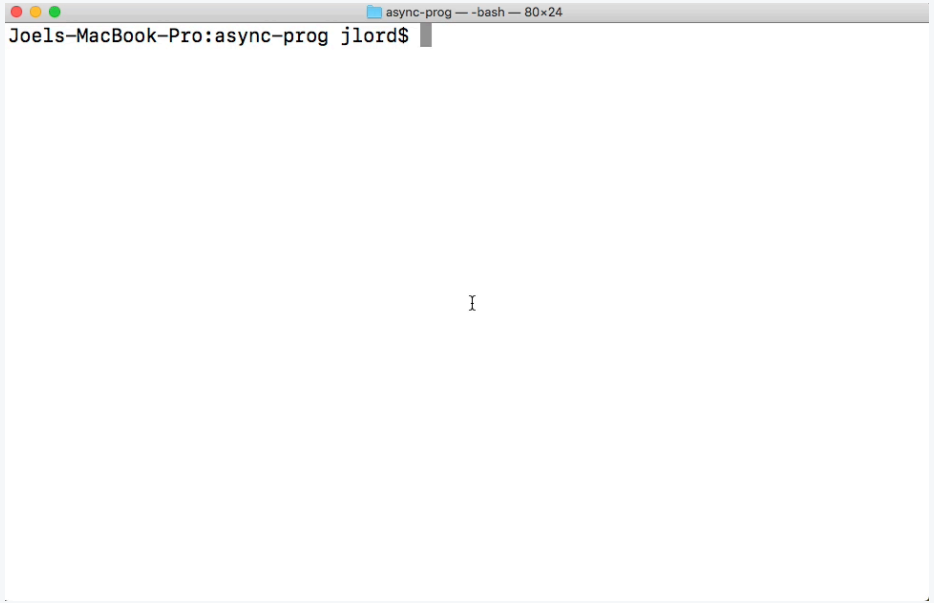


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

Result

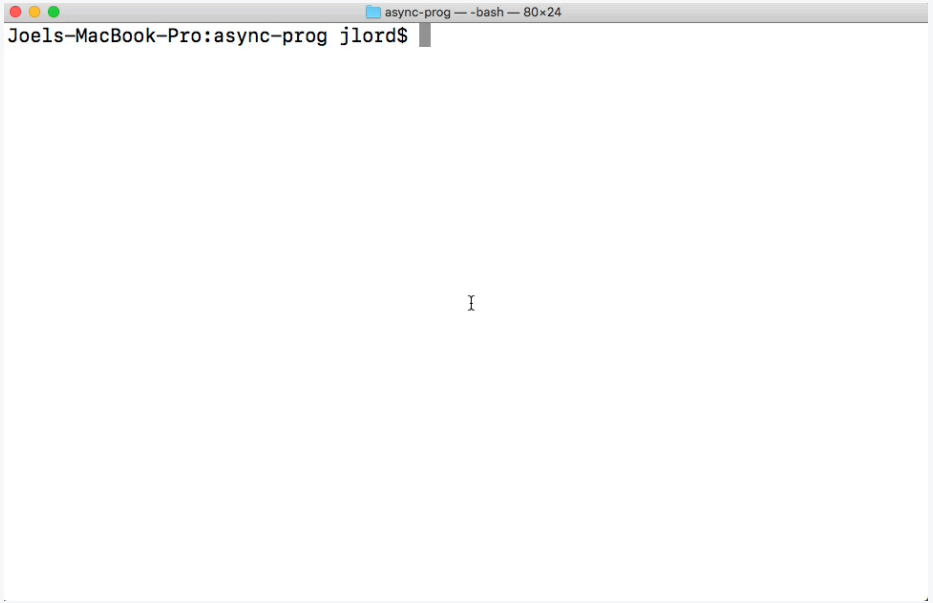


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

Result

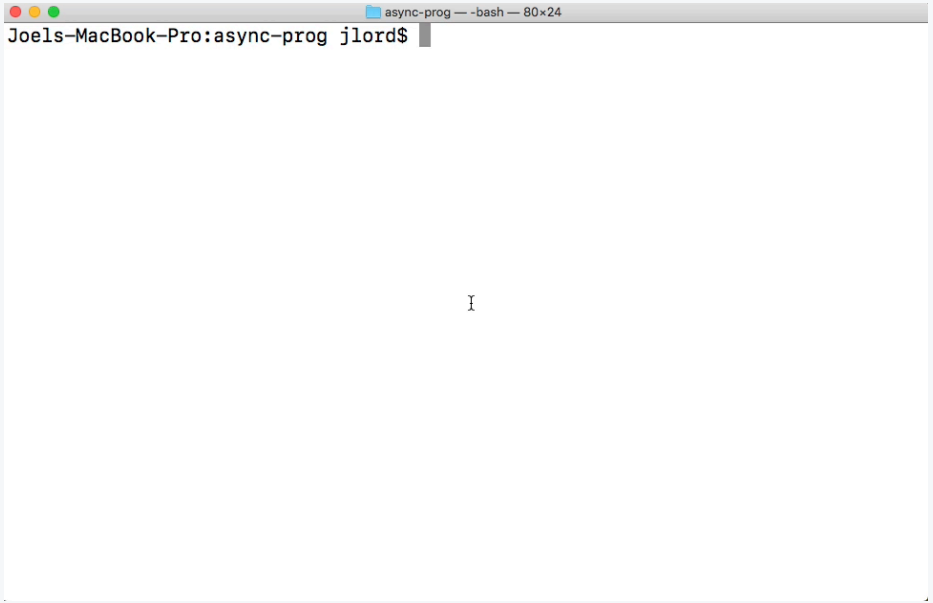


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

Result

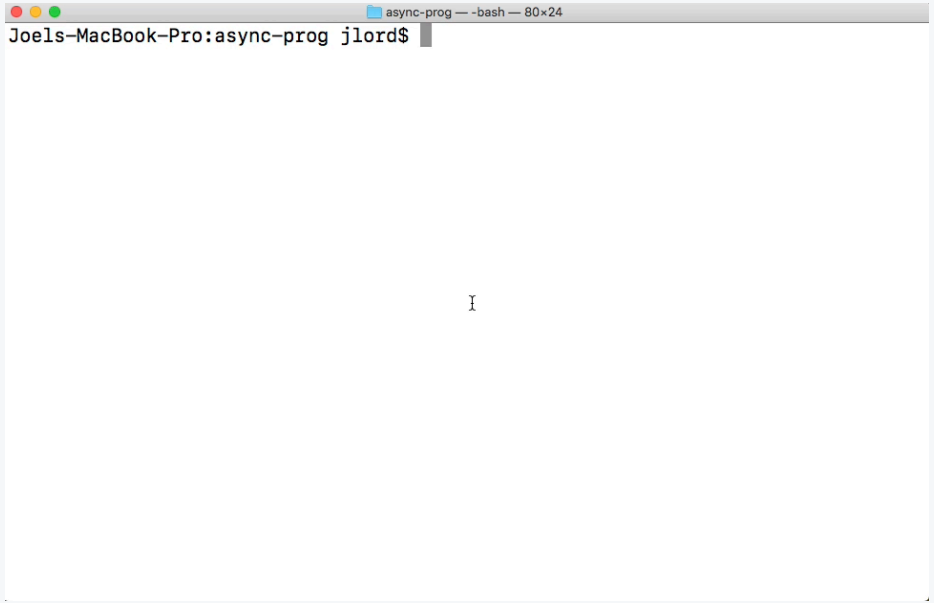


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

Result

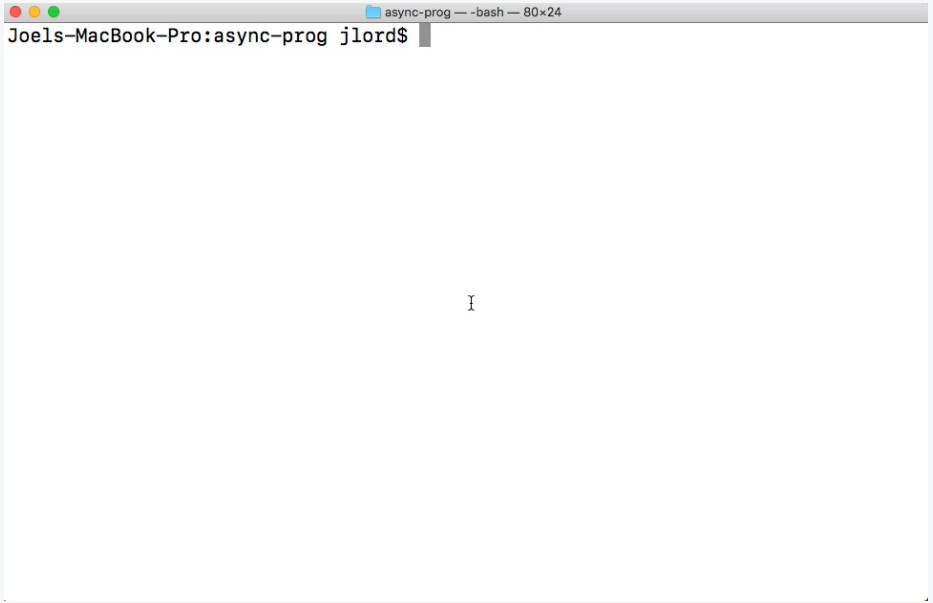


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
});
```

Result

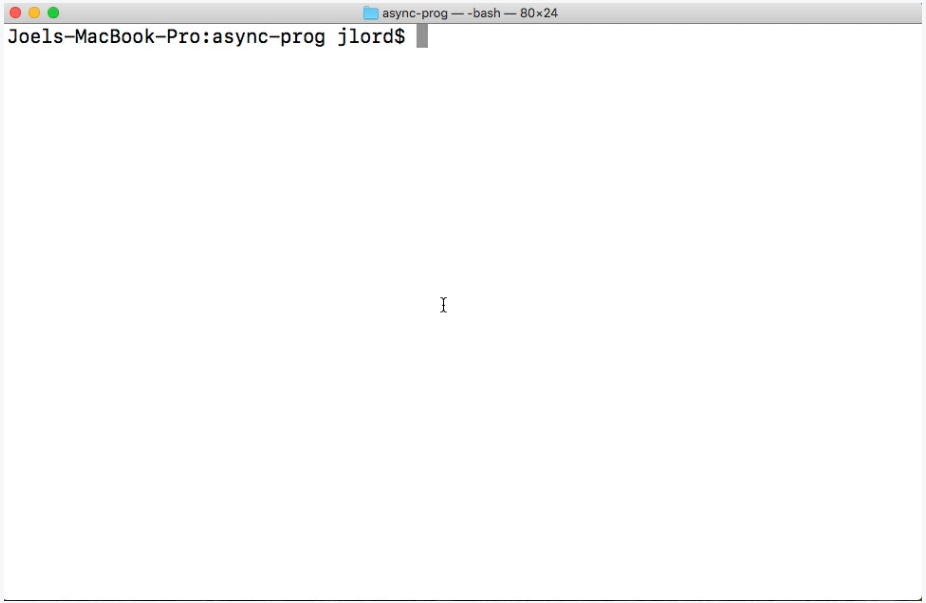


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.simulateError();

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).catch((err) => {
    console.log("Error fetching the ingredients");
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).catch((err) => {
    console.log("An error occurred while preparing the sandwich");
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
}).catch((err) => {
    console.log("An error occurred");
});
```

Result

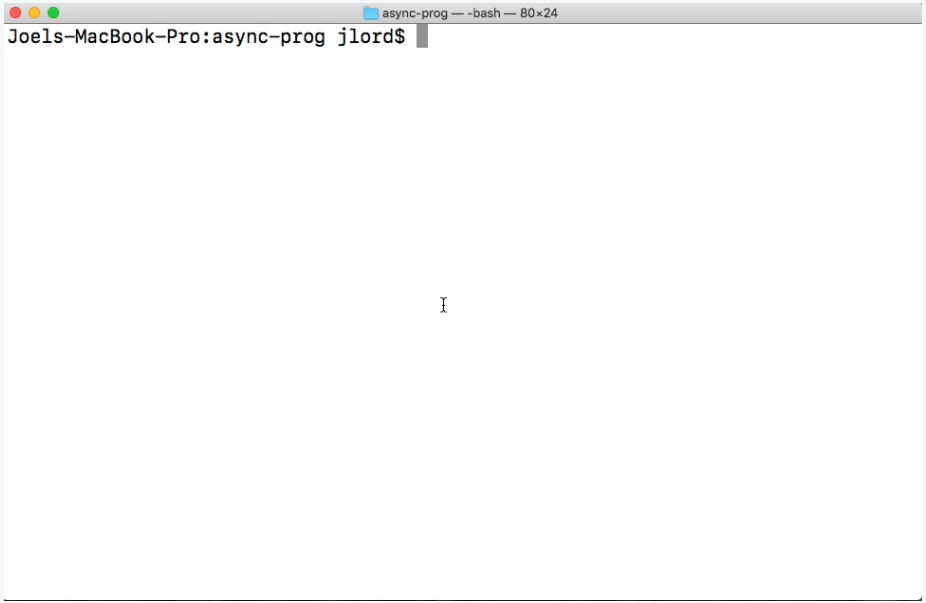


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.simulateError();

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).catch((err) => {
    console.log("Error fetching the ingredients");
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).catch((err) => {
    console.log("An error occurred while preparing the sandwich");
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
}).catch((err) => {
    console.log("An error occurred");
});
```

Result

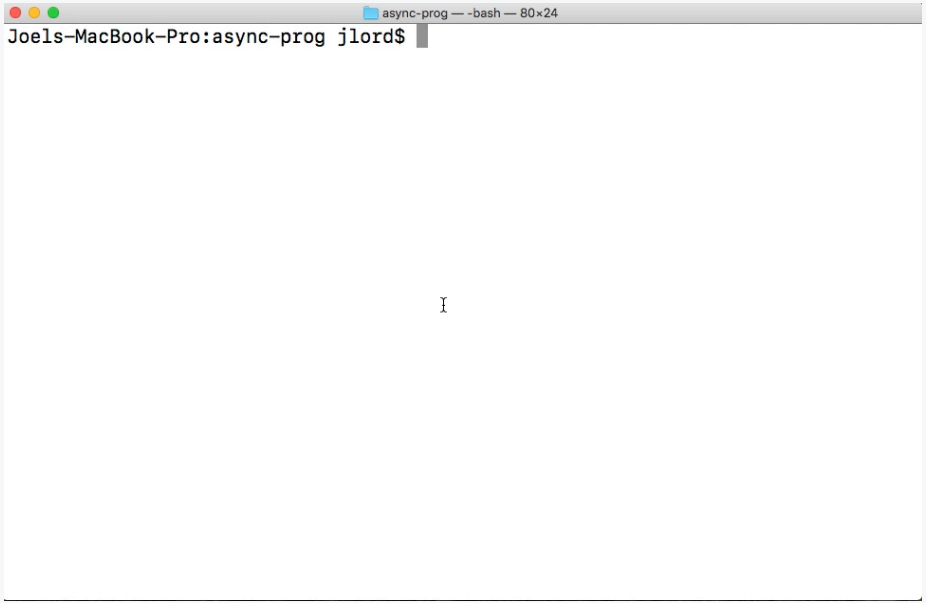


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.simulateError();

PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
}).catch((err) => {
    console.log("Error fetching the ingredients");
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
}).then(() => {
    return PBnJMaker.spreadJam();
}).catch((err) => {
    console.log("An error occurred while preparing the sandwich");
}).then(() => {
    return PBnJMaker.closeSandwich();
}).then((result) => {
    console.log("Eat that " + result);
}).catch((err) => {
    console.log("An error occurred");
});
```

Result



@joel__lord
#confoo

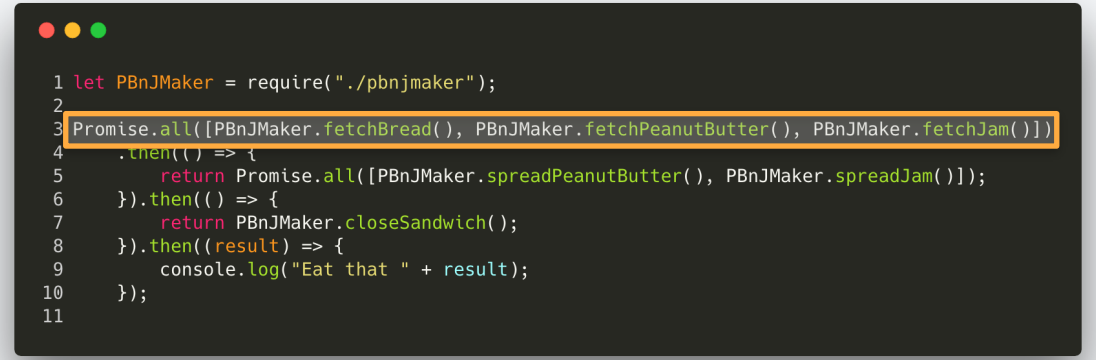
Aggregate functions

- .all()
- .race()

```
let PBnJMaker = require("./pbnjmaker");

Promise.all([PBnJMaker.fetchBread(), PBnJMaker.fetchPeanutButter(), PBnJMaker.fetchJam()])
    .then(() => {
        return Promise.all([PBnJMaker.spreadPeanutButter(), PBnJMaker.spreadJam()]);
    }).then(() => {
        return PBnJMaker.closeSandwich();
    }).then((result) => {
        console.log("Eat that " + result);
    });

```

Aggregate functions

- .all()
- .race()



```
let PBnJMaker = require("./pbnjmaker");

Promise.all([PBnJMaker.fetchBread(), PBnJMaker.fetchPeanutButter(), PBnJMaker.fetchJam()])
    .then(() => {
        return Promise.all([PBnJMaker.spreadPeanutButter(), PBnJMaker.spreadJam()]);
    }).then(() => {
        return PBnJMaker.closeSandwich();
    }).then((result) => {
        console.log("Eat that " + result);
    });

```

Aggregate functions

- .all()
- .race()

```
let PBnJMaker = require("./pbnjmaker");

Promise.all([PBnJMaker.fetchBread(), PBnJMaker.fetchPeanutButter(), PBnJMaker.fetchJam()])
    .then(() => {
        return Promise.all([PBnJMaker.spreadPeanutButter(), PBnJMaker.spreadJam()]);
    }).then(() => {
        return PBnJMaker.closeSandwich();
    }).then((result) => {
        console.log("Eat that " + result);
    });

```

Aggregate functions

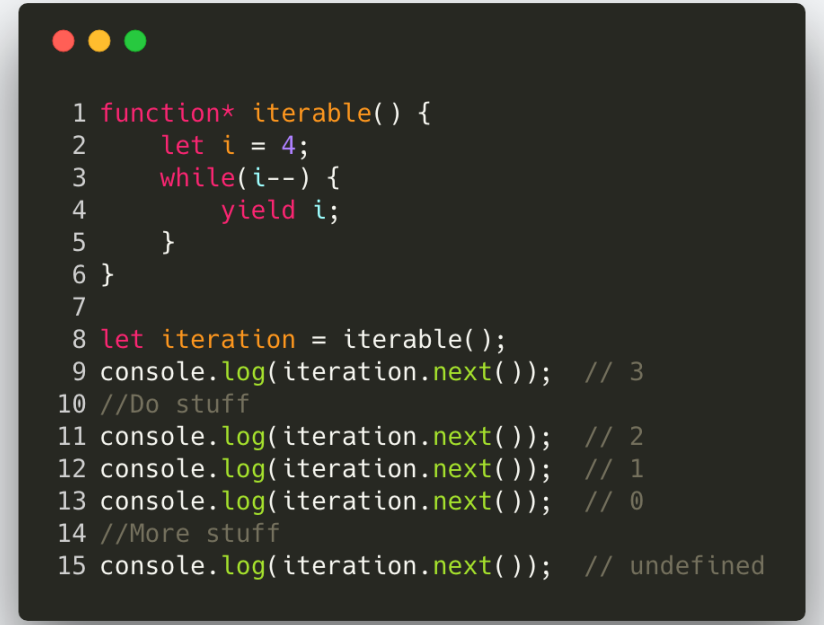
- .all()
- .race()

```
let PBnJMaker = require("./pbnjmaker");

Promise.all([PBnJMaker.fetchBread(), PBnJMaker.fetchPeanutButter(), PBnJMaker.fetchJam()])
    .then(() => {
        return Promise.all([PBnJMaker.spreadPeanutButter(), PBnJMaker.spreadJam()]);
    }).then(() => {
        return PBnJMaker.closeSandwich();
    }).then((result) => {
        console.log("Eat that " + result);
    });

```


Generators
(2015)

What is a generator?

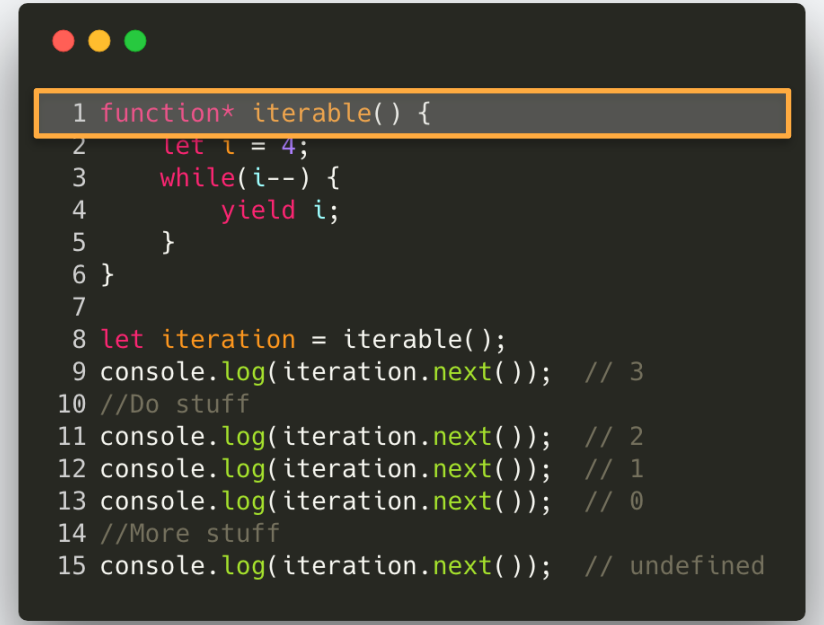
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

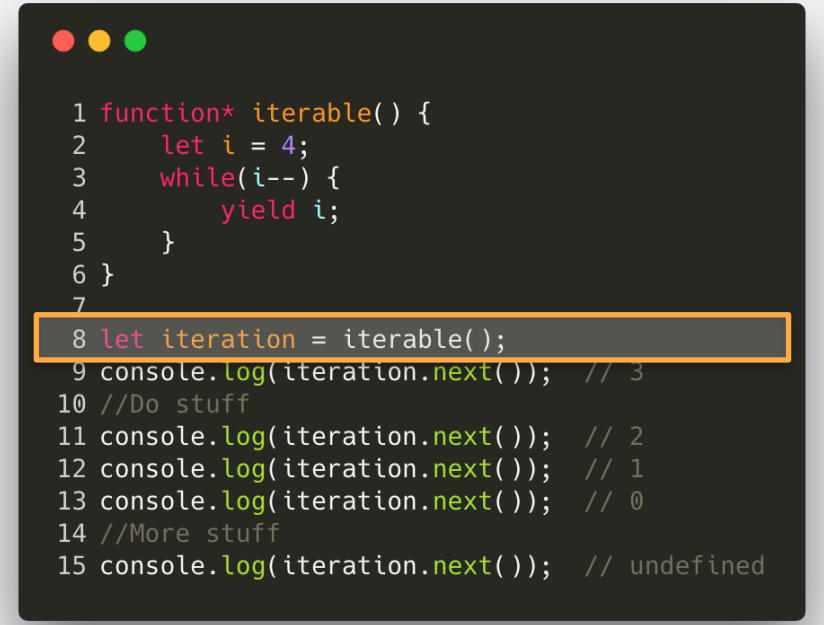
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

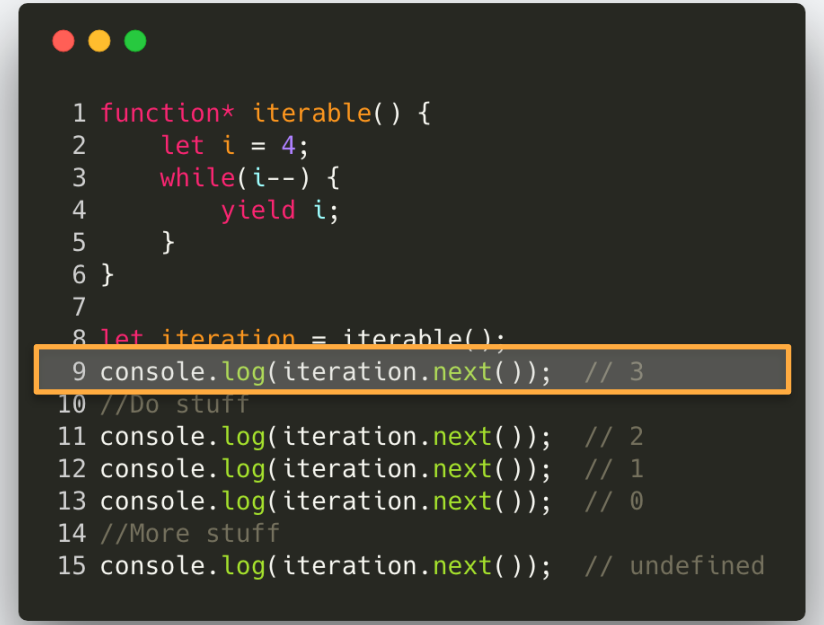
- An iterable function
- “Pauses” the execution of a function



```
1 function* iterable() {
2     let i = 4;
3     while(i--) {
4         yield i;
5     }
6 }
7
8 let iteration = iterable();
9 console.log(iteration.next());  // 3
10 //Do stuff
11 console.log(iteration.next());  // 2
12 console.log(iteration.next());  // 1
13 console.log(iteration.next());  // 0
14 //More stuff
15 console.log(iteration.next());  // undefined
```

What is a generator?

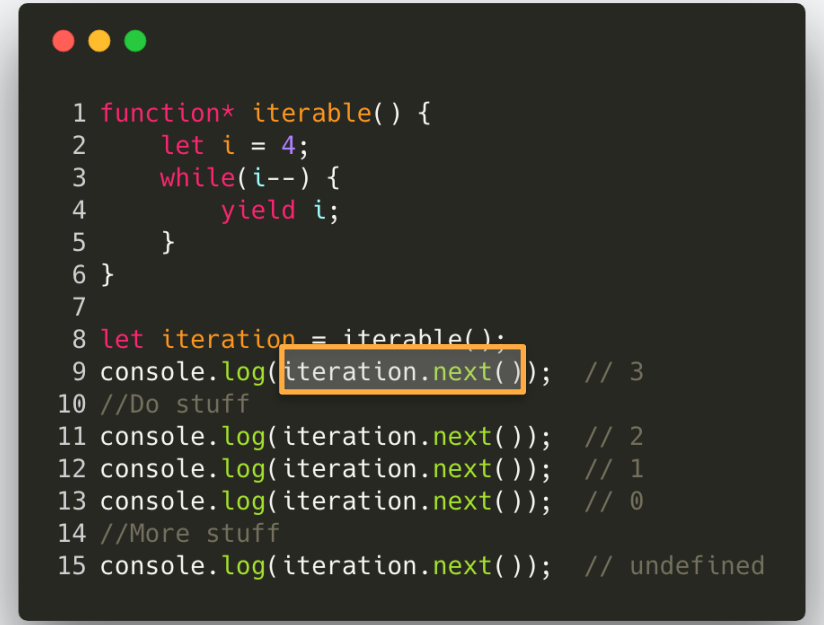
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

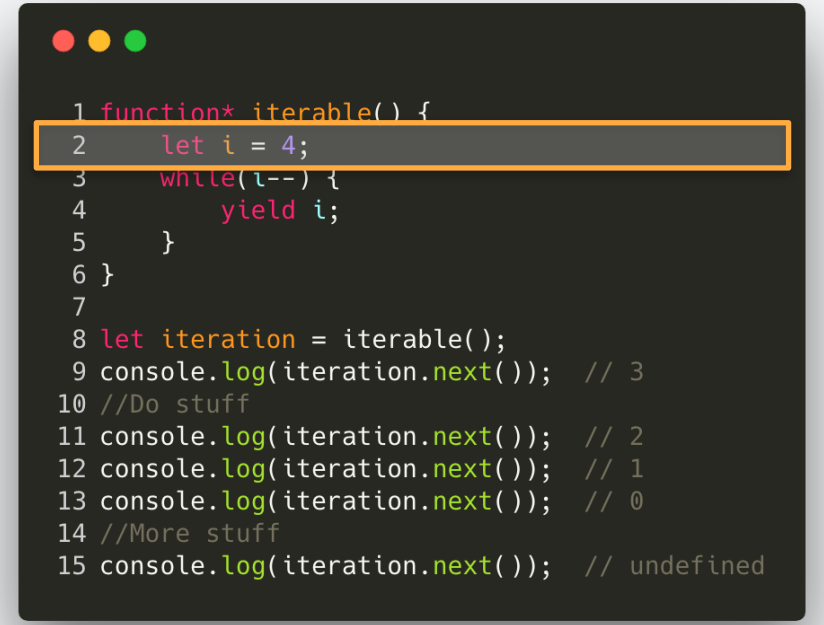
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

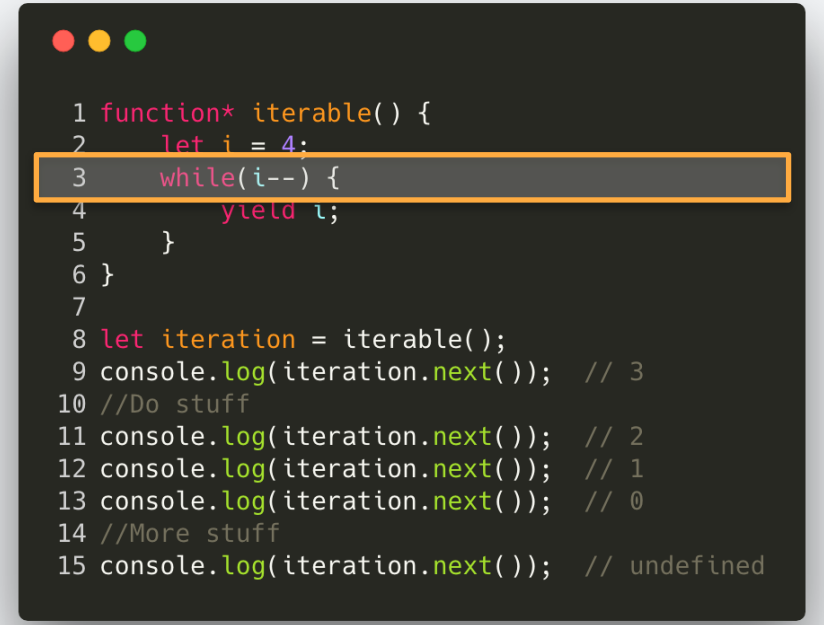
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

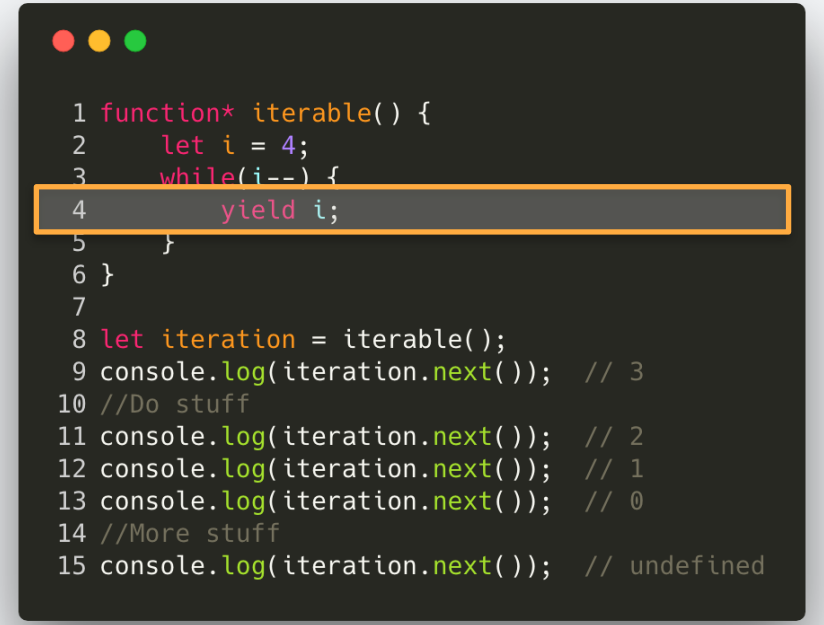
- An iterable function
- “Pauses” the execution of a function



```
1 function* iterable() {
2     let i = 4;
3     while(i--) {
4         yield i;
5     }
6 }
7
8 let iteration = iterable();
9 console.log(iteration.next());  // 3
10 //Do stuff
11 console.log(iteration.next());  // 2
12 console.log(iteration.next());  // 1
13 console.log(iteration.next());  // 0
14 //More stuff
15 console.log(iteration.next());  // undefined
```

What is a generator?

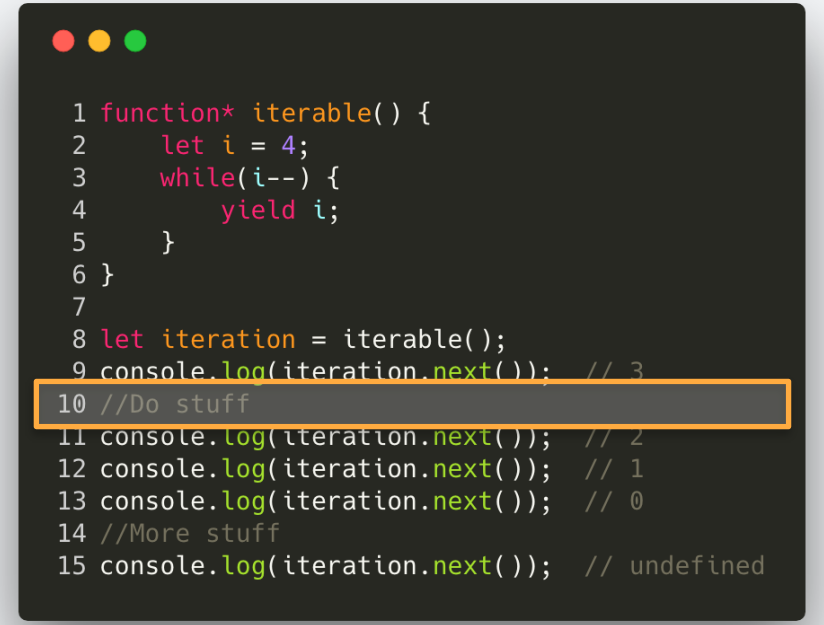
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

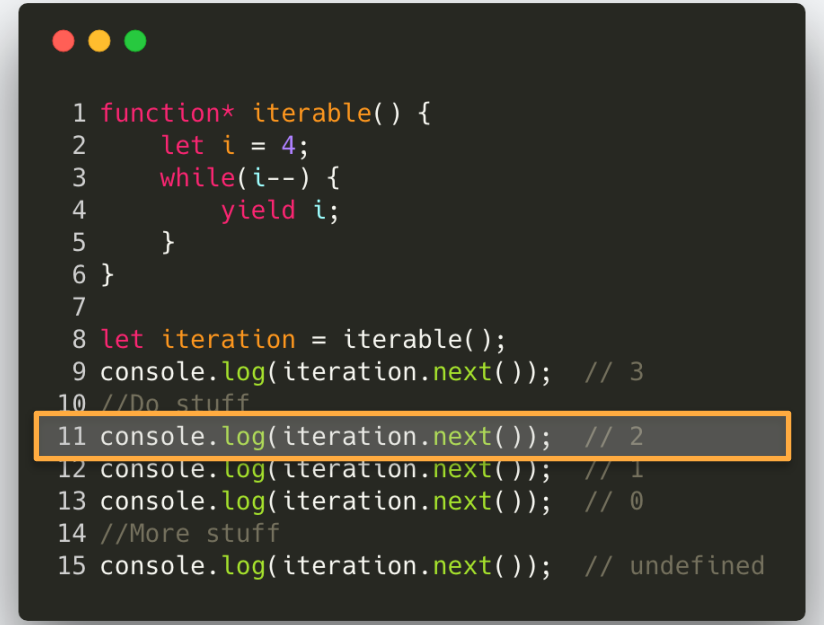
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

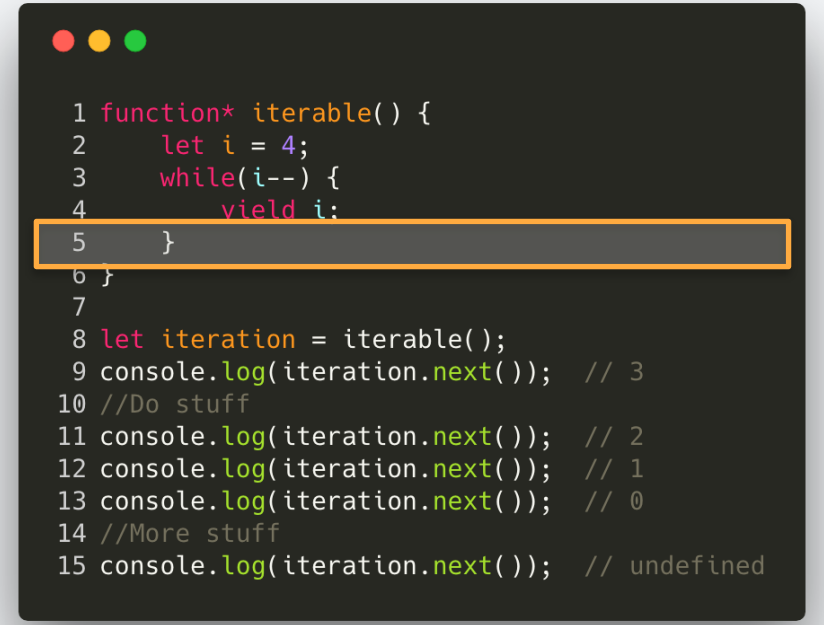
- An iterable function
- “Pauses” the execution of a function



```
1 function* iterable() {
2     let i = 4;
3     while(i--) {
4         yield i;
5     }
6 }
7
8 let iteration = iterable();
9 console.log(iteration.next());  // 3
10 //Do stuff
11 console.log(iteration.next());  // 2
12 console.log(iteration.next());  // 1
13 console.log(iteration.next());  // 0
14 //More stuff
15 console.log(iteration.next());  // undefined
```

What is a generator?

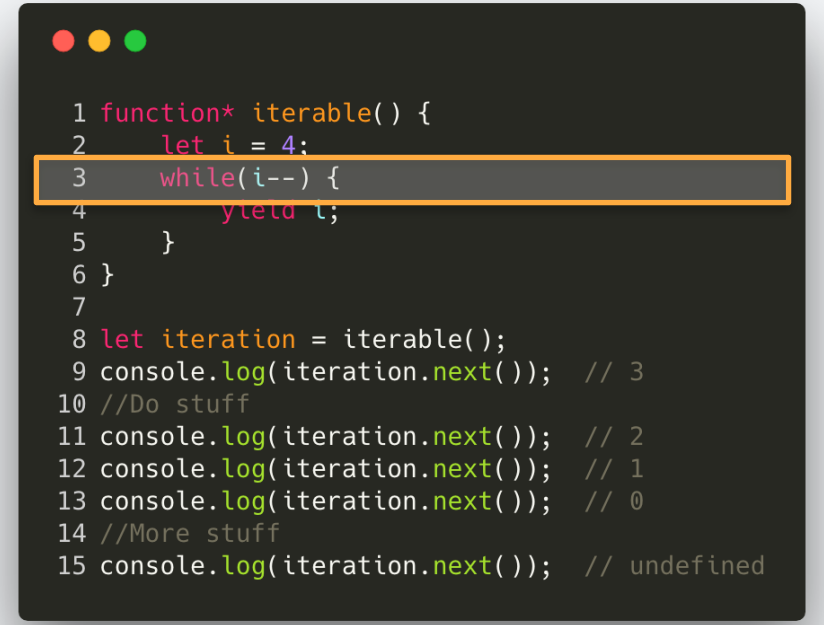
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

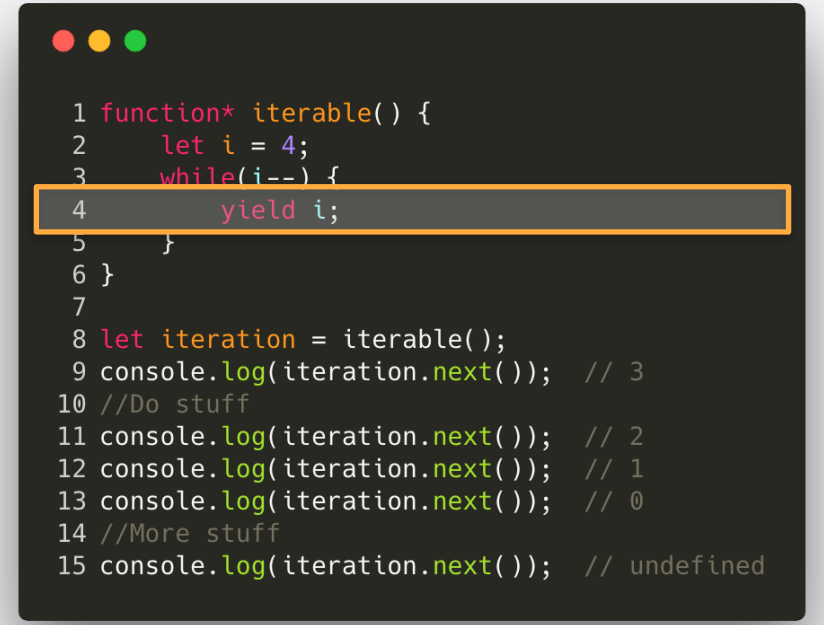
- An iterable function
- “Pauses” the execution of a function



```
1 function* iterable() {
2     let i = 4;
3     while(i--) {
4         yield i;
5     }
6 }
7
8 let iteration = iterable();
9 console.log(iteration.next());  // 3
10 //Do stuff
11 console.log(iteration.next());  // 2
12 console.log(iteration.next());  // 1
13 console.log(iteration.next());  // 0
14 //More stuff
15 console.log(iteration.next());  // undefined
```

What is a generator?

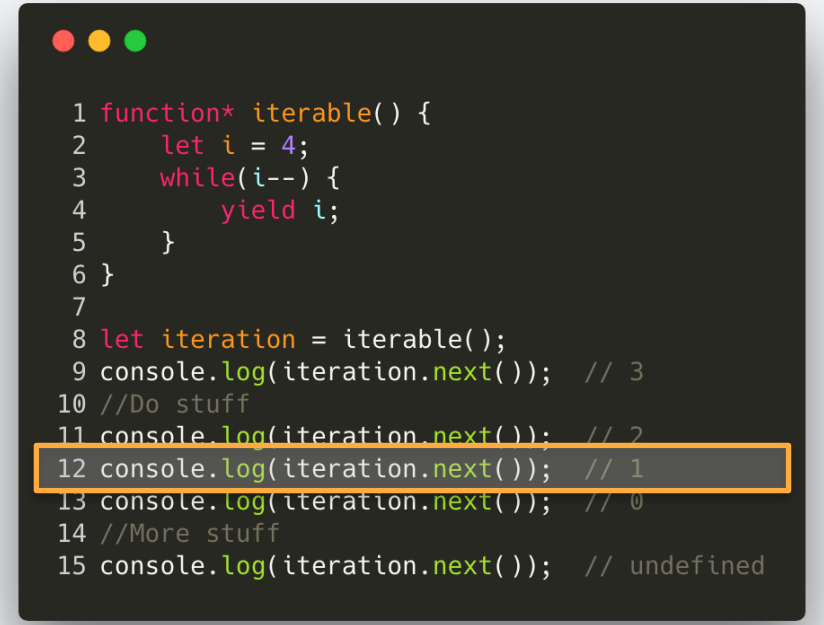
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

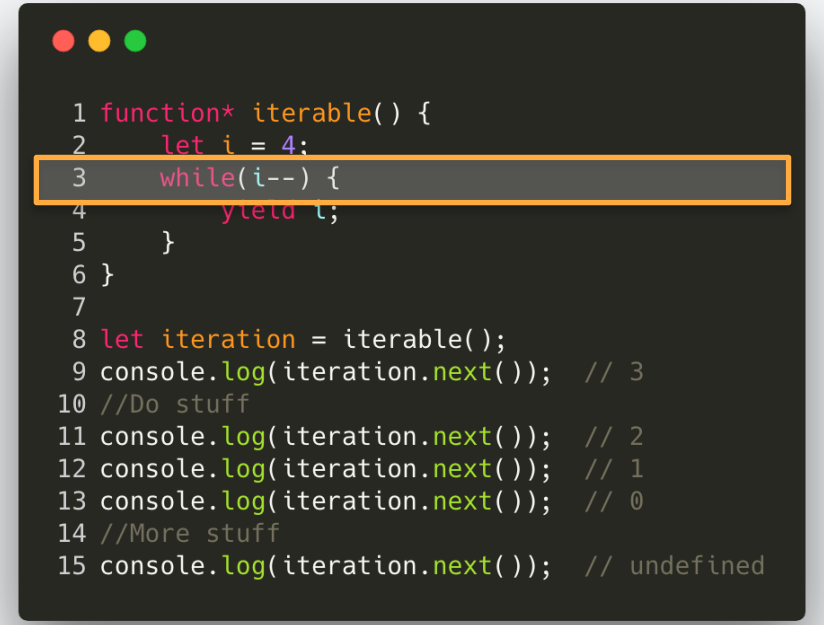
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

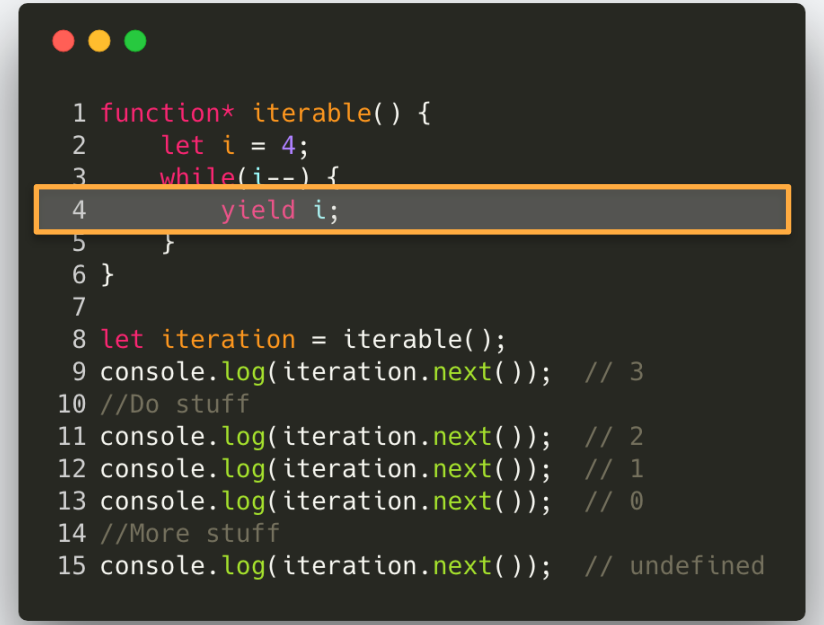
- An iterable function
- “Pauses” the execution of a function



```
1 function* iterable() {
2     let i = 4;
3     while(i--) {
4         yield i;
5     }
6 }
7
8 let iteration = iterable();
9 console.log(iteration.next());  // 3
10 //Do stuff
11 console.log(iteration.next());  // 2
12 console.log(iteration.next());  // 1
13 console.log(iteration.next());  // 0
14 //More stuff
15 console.log(iteration.next());  // undefined
```

What is a generator?

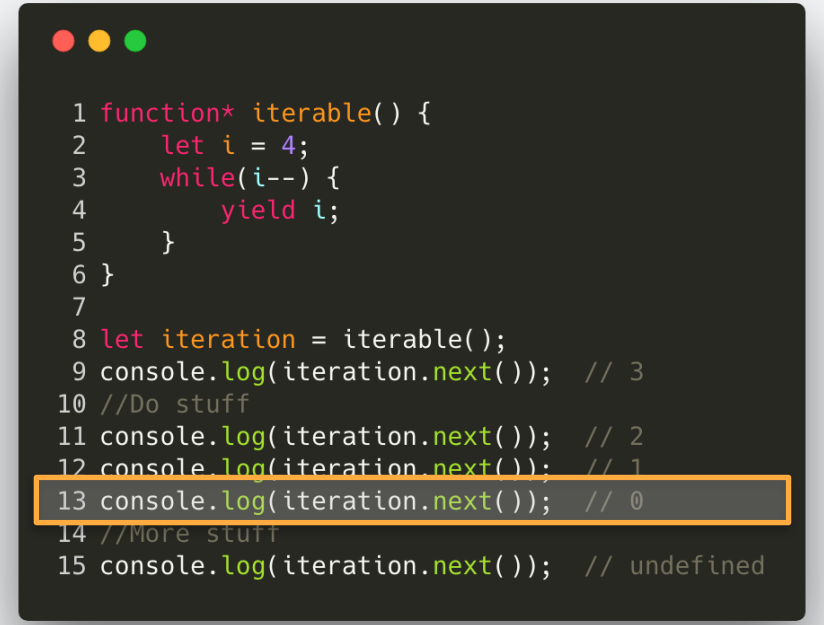
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

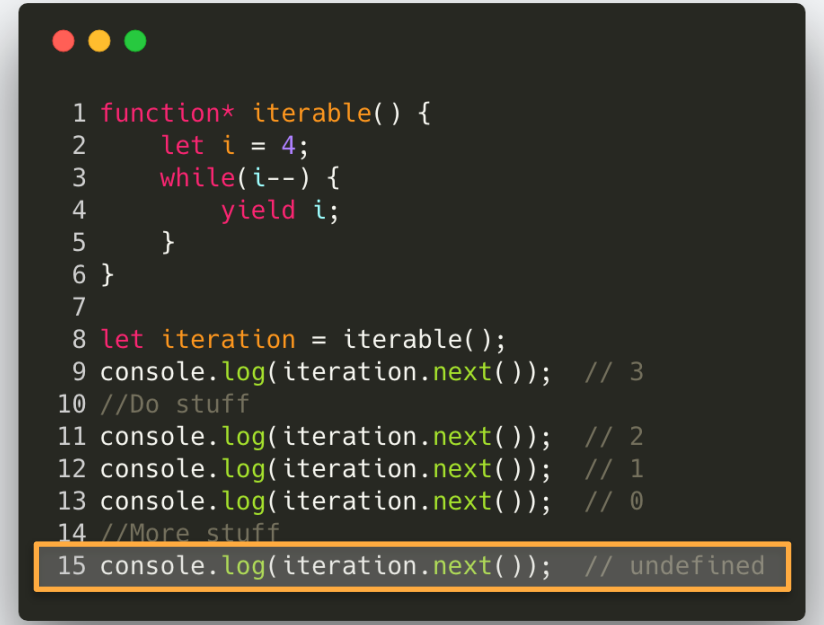
- An iterable function
- “Pauses” the execution of a function



```
 1 function* iterable() {
 2     let i = 4;
 3     while(i--) {
 4         yield i;
 5     }
 6 }
 7
 8 let iteration = iterable();
 9 console.log(iteration.next());  // 3
10 //Do stuff
11 console.log(iteration.next());  // 2
12 console.log(iteration.next());  // 1
13 console.log(iteration.next());  // 0
14 //More stuff
15 console.log(iteration.next());  // undefined
```

What is a generator?

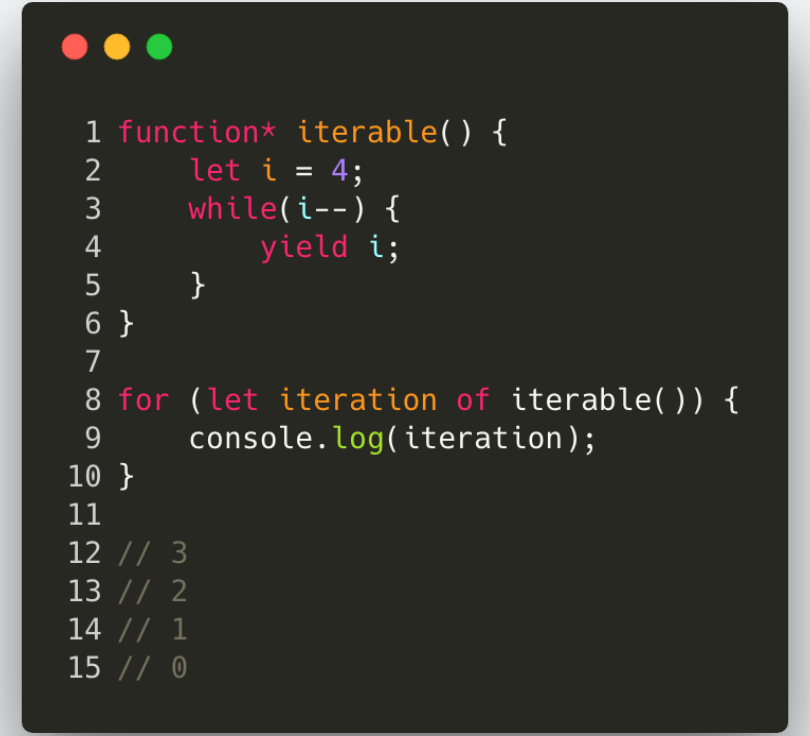
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

let iteration = iterable();
console.log(iteration.next());  // 3
//Do stuff
console.log(iteration.next());  // 2
console.log(iteration.next());  // 1
console.log(iteration.next());  // 0
//More stuff
console.log(iteration.next());  // undefined
```

What is a generator?

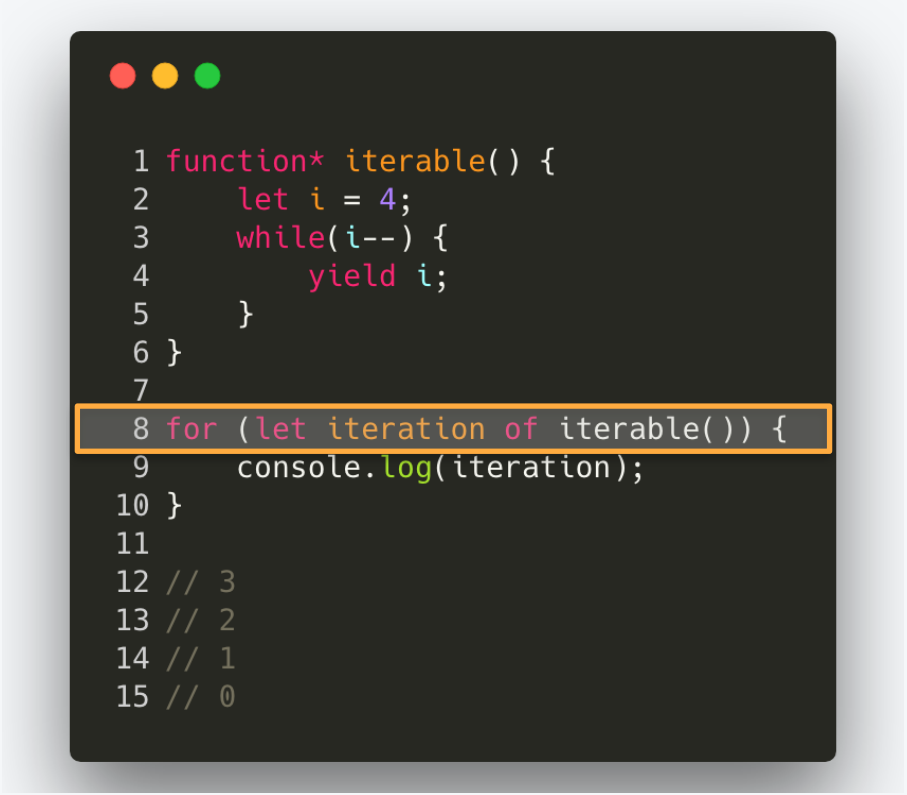
- An iterable function
- “Pauses” the execution of a function



```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

for (let iteration of iterable()) {
    console.log(iteration);
}

// 3
// 2
// 1
// 0
```

What is a generator?

- An iterable function
- “Pauses” the execution of a function

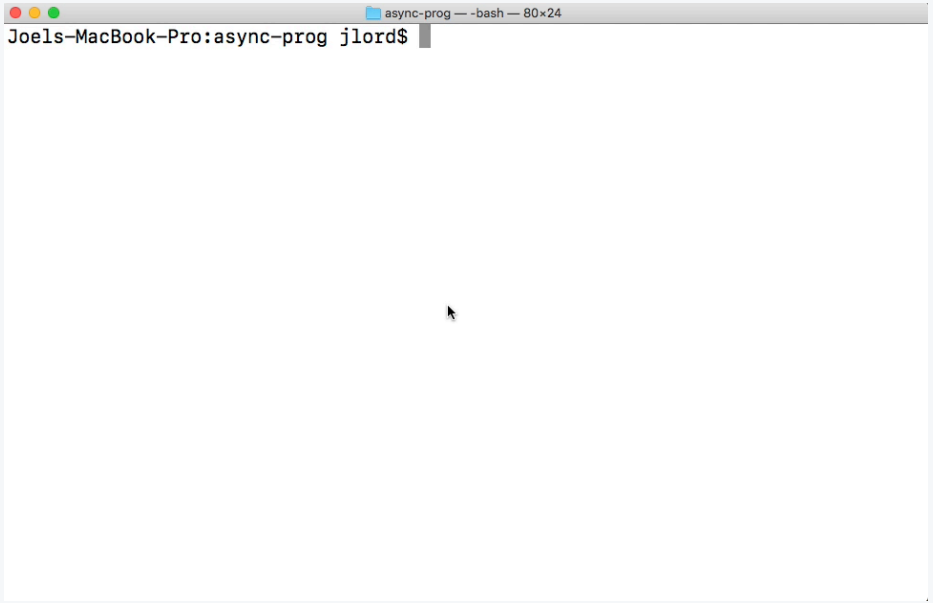


```
function* iterable() {
    let i = 4;
    while(i--) {
        yield i;
    }
}

for (let iteration of iterable()) {
    console.log(iteration);
}

// 3
// 2
// 1
// 0
```

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.verbose();

function* makePBnJ() {
    yield PBnJMaker.fetchBread;
    yield PBnJMaker.fetchPeanutButter;
    yield PBnJMaker.fetchJam;
    yield PBnJMaker.spreadPeanutButter;
    yield PBnJMaker.spreadJam;
    yield PBnJMaker.closeSandwich;
}

for (let step of makePBnJ()) {
    step();
}

console.log("Eat that sandwich");
```

Result



@joel__lord
#confoo

Async Generators

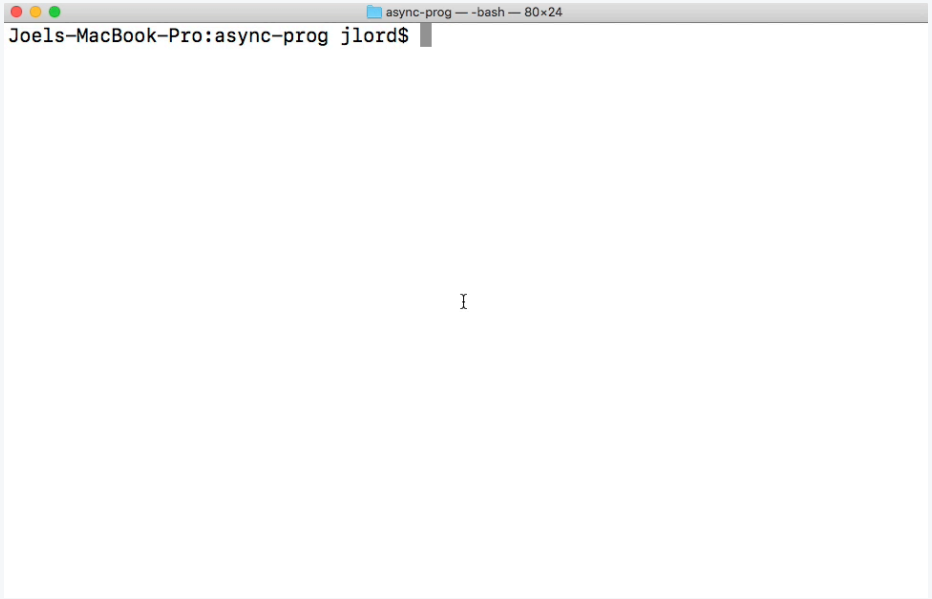
- Combining the power of promises and generators

```
module.exports = (makeGenerator) => {
    let generator = makeGenerator.apply(this, arguments);

    function handle(result) {
        // A generator returns an object
        // {done: [Boolean], value: [Any]}
        if (result.done) return Promise.resolve(result.value);
        return Promise.resolve(result.value).then(function(res) {
            return handle(generator.next(res));
        }, function(err) {
            return handle(generator.throw(err));
        });
    }

    try {
        return handle(generator.next());
    } catch(e) {
        return Promise.reject(e);
    }
};
```

Code

```
let PBnJMaker = require("./pbnjmaker");
let async = require("./async");

function* makePBnJ() {
    yield PBnJMaker.fetchBread();
    yield PBnJMaker.fetchPeanutButter();
    yield PBnJMaker.fetchJam();
    yield PBnJMaker.spreadPeanutButter();
    yield PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

async(makePBnJ).then((result) => console.log("Eat that " + result));
```

Result

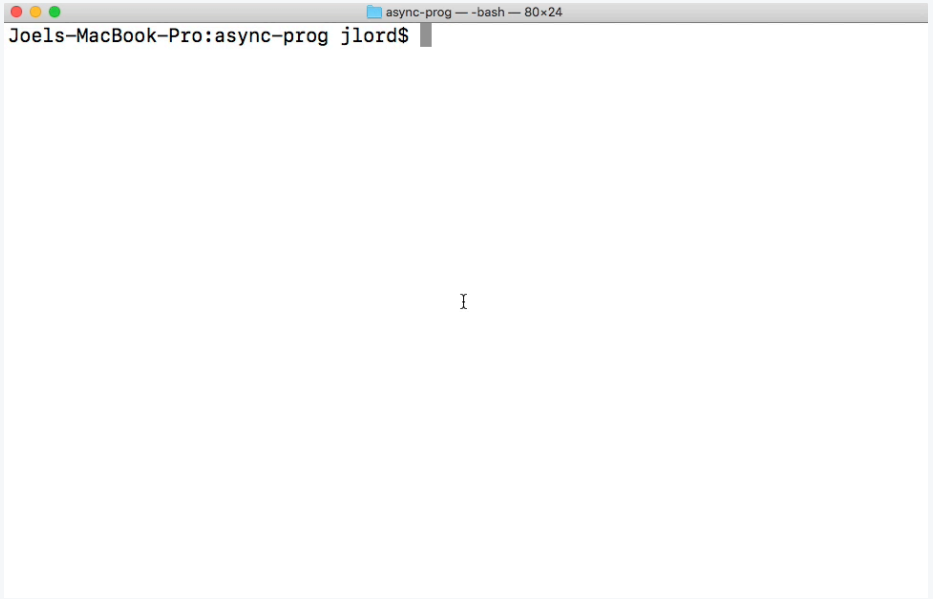


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");
let async = require("./async");

function* makePBnJ() {
    yield PBnJMaker.fetchBread();
    yield PBnJMaker.fetchPeanutButter();
    yield PBnJMaker.fetchJam();
    yield PBnJMaker.spreadPeanutButter();
    yield PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

async(makePBnJ).then((result) => console.log("Eat that " + result));
```

Result



@joel__lord
#confoo


Async / Await
(2017)

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

```
let PBnJMaker = require("./pbnjmaker");

async function makePBnJ() {
    await PBnJMaker.fetchBread();
    await PBnJMaker.fetchPeanutButter();
    await PBnJMaker.fetchJam();
    await PBnJMaker.spreadPeanutButter();
    await PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

```
let PBnJMaker = require("./pbnjmaker");

async function makePBnJ() {
    await PBnJMaker.fetchBread();
    await PBnJMaker.fetchPeanutButter();
    await PBnJMaker.fetchJam();
    await PBnJMaker.spreadPeanutButter();
    await PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

```
let PBnJMaker = require("./pbnjmaker");

async function makePBnJ() {
    await PBnJMaker.fetchBread();
    await PBnJMaker.fetchPeanutButter();
    await PBnJMaker.fetchJam();
    await PBnJMaker.spreadPeanutButter();
    await PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

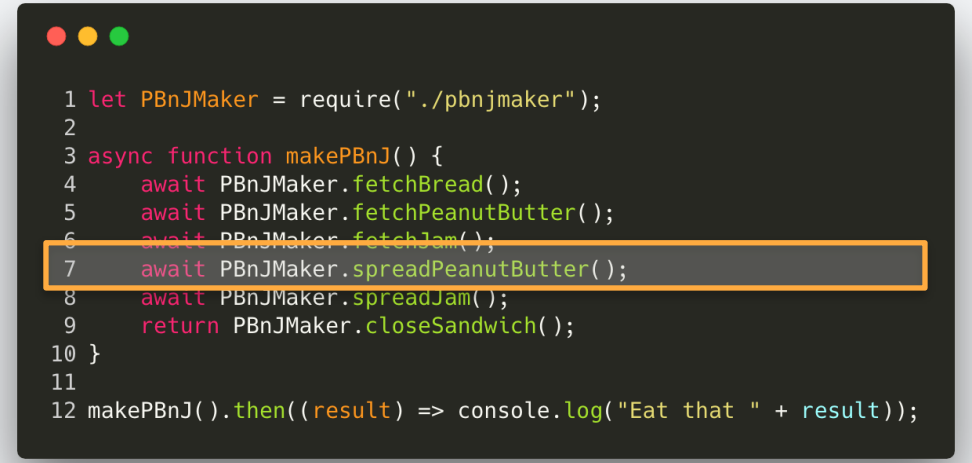
What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

```
let PBnJMaker = require("./pbnjmaker");

async function makePBnJ() {
    await PBnJMaker.fetchBread();
    await PBnJMaker.fetchPeanutButter();
    await PBnJMaker.fetchJam();
    await PBnJMaker.spreadPeanutButter();
    await PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises



```
let PBnJMaker = require("./pbnjmaker");

async function makePBnJ() {
    await PBnJMaker.fetchBread();
    await PBnJMaker.fetchPeanutButter();
    await PBnJMaker.fetchJam();
    await PBnJMaker.spreadPeanutButter();
    await PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

```
let PBnJMaker = require("./pbnjmaker");

async function makePBnJ() {
    await PBnJMaker.fetchBread();
    await PBnJMaker.fetchPeanutButter();
    await PBnJMaker.fetchJam();
    await PBnJMaker.spreadPeanutButter();
    await PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

```
let PBnJMaker = require("./pbnjmaker");

async function makePBnJ() {
    await PBnJMaker.fetchBread();
    await PBnJMaker.fetchPeanutButter();
    await PBnJMaker.fetchJam();
    await PBnJMaker.spreadPeanutButter();
    await PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Newly introduced in ES8 (ES2017)
- Natural evolution of generators and promises

```
let PBnJMaker = require("./pbnjmaker");

async function makePBnJ() {
    await PBnJMaker.fetchBread();
    await PBnJMaker.fetchPeanutButter();
    await PBnJMaker.fetchJam();
    await PBnJMaker.spreadPeanutButter();
    await PBnJMaker.spreadJam();
    return PBnJMaker.closeSandwich();
}

makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Very easy to parallelize processes

```
let PBnJMaker = require("./pbnjmaker");

async function getIngredients() {
    let p1 = PBnJMaker.fetchBread();
    let p2 = PBnJMaker.fetchPeanutButter();
    let p3 = PBnJMaker.fetchJam();
    return await p1 + await p2 + await p3;
}

async function prepareSandwich() {
    let p1 = PBnJMaker.spreadPeanutButter();
    let p2 = PBnJMaker.spreadJam();
    return await p1 + await p2;
}

async function makePBnJ() {
    await getIngredients();
    await prepareSandwich();
    return PBnJMaker.closeSandwich();
}
makePBnJ().then((result) => console.log("Eat that " + result));

```

What is async/await?

- Very easy to parallelize processes

```
let PBnJMaker = require("./pbnjmaker");

async function getIngredients() {
    let p1 = PBnJMaker.fetchBread();
    let p2 = PBnJMaker.fetchPeanutButter();
    let p3 = PBnJMaker.fetchJam();
    return await p1 + await p2 + await p3;
}

async function prepareSandwich() {
    let p1 = PBnJMaker.spreadPeanutButter();
    let p2 = PBnJMaker.spreadJam();
    return await p1 + await p2;
}

async function makePBnJ() {
    await getIngredients();
    await prepareSandwich();
    return PBnJMaker.closeSandwich();
}
makePBnJ().then((result) => console.log("Eat that " + result));

```

What is async/await?

- Very easy to parallelize processes

```
let PBnJMaker = require("./pbnjmaker");

async function getIngredients() {
    let p1 = PBnJMaker.fetchBread();
    let p2 = PBnJMaker.fetchPeanutButter();
    let p3 = PBnJMaker.fetchJam();
    return await p1 + await p2 + await p3;
}

async function prepareSandwich() {
    let p1 = PBnJMaker.spreadPeanutButter();
    let p2 = PBnJMaker.spreadJam();
    return await p1 + await p2;
}

async function makePBnJ() {
    await getIngredients();
    await prepareSandwich();
    return PBnJMaker.closeSandwich();
}
makePBnJ().then((result) => console.log("Eat that " + result));

```

What is async/await?

- Very easy to parallelize processes

```
let PBnJMaker = require("./pbnjmaker");

async function getIngredients() {
    let p1 = PBnJMaker.fetchBread();
    let p2 = PBnJMaker.fetchPeanutButter();
    let p3 = PBnJMaker.fetchJam();
    return await p1 + await p2 + await p3;
}

async function prepareSandwich() {
    let p1 = PBnJMaker.spreadPeanutButter();
    let p2 = PBnJMaker.spreadJam();
    return await p1 + await p2;
}

async function makePBnJ() {
    await getIngredients();
    await prepareSandwich();
    return PBnJMaker.closeSandwich();
}
makePBnJ().then((result) => console.log("Eat that " + result));

```

What is async/await?

- Very easy to parallelize processes

```
let PBnJMaker = require("./pbnjmaker");

async function getIngredients() {
    let p1 = PBnJMaker.fetchBread();
    let p2 = PBnJMaker.fetchPeanutButter();
    let p3 = PBnJMaker.fetchJam();
    return await p1 + await p2 + await p3;
}

async function prepareSandwich() {
    let p1 = PBnJMaker.spreadPeanutButter();
    let p2 = PBnJMaker.spreadJam();
    return await p1 + await p2;
}

async function makePBnJ() {
    await getIngredients();
    await prepareSandwich();
    return PBnJMaker.closeSandwich();
}
makePBnJ().then((result) => console.log("Eat that " + result));

```

What is async/await?

- Very easy to parallelize processes

```
let PBnJMaker = require("./pbnjmaker");

async function getIngredients() {
    let p1 = PBnJMaker.fetchBread();
    let p2 = PBnJMaker.fetchPeanutButter();
    let p3 = PBnJMaker.fetchJam();
    return await p1 + await p2 + await p3;
}

async function prepareSandwich() {
    let p1 = PBnJMaker.spreadPeanutButter();
    let p2 = PBnJMaker.spreadJam();
    return await p1 + await p2;
}

async function makePBnJ() {
    await getIngredients();
    await prepareSandwich();
    return PBnJMaker.closeSandwich();
}
makePBnJ().then((result) => console.log("Eat that " + result));
```

What is async/await?

- Very easy to parallelize processes

```
let PBnJMaker = require("./pbnjmaker");

async function getIngredients() {
    let p1 = PBnJMaker.fetchBread();
    let p2 = PBnJMaker.fetchPeanutButter();
    let p3 = PBnJMaker.fetchJam();
    return await p1 + await p2 + await p3;
}

async function prepareSandwich() {
    let p1 = PBnJMaker.spreadPeanutButter();
    let p2 = PBnJMaker.spreadJam();
    return await p1 + await p2;
}

async function makePBnJ() {
    await getIngredients();
    await prepareSandwich();
    return PBnJMaker.closeSandwich();
}
makePBnJ().then((result) => console.log("Eat that " + result));

```

Async / Await

Programatically define your steps

```
let PBnJMaker = require("./pbnjmaker");

let ingredients = ["Bread", "PeanutButter", "Jam"];
let steps = ingredients.map((i) => `fetch${i}`);
steps.push(...["spreadPeanutButter", "spreadJam"]);

async function makePBnJ(todo) {
    for (var i = 0; i < todo.length; i++) {
        await PBnJMaker[todo[i]]();
    }
    return PBnJMaker.closeSandwich();
}

makePBnJ(steps).then((result) => console.log("Eat that " + result));
```

Events (1990's)

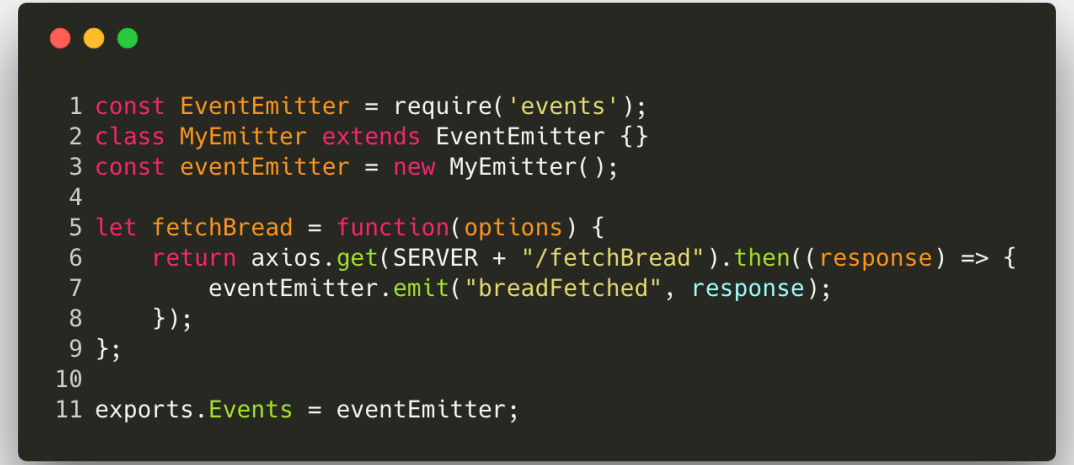
Events in the browser

- At the core of the original Javascript
- window.onready = …
- button.onclick = …

```html
<html>
<body>
    <button id="btn1">Click me</button>
</body>

<script>
    let btn = document.querySelector("#btn1");
    btn.onclick = () => alert("Clicked");
</script>
</html>
```

Node Event Emitter

- In node, you can use the event emitter
- Create you own events for a library



```
const EventEmitter = require('events');
class MyEmitter extends EventEmitter {}
const eventEmitter = new MyEmitter();

let fetchBread = function(options) {
    return axios.get(SERVER + "/fetchBread").then((response) => {
        eventEmitter.emit("breadFetched", response);
    });
};

exports.Events = eventEmitter;
```

Node Event Emitter

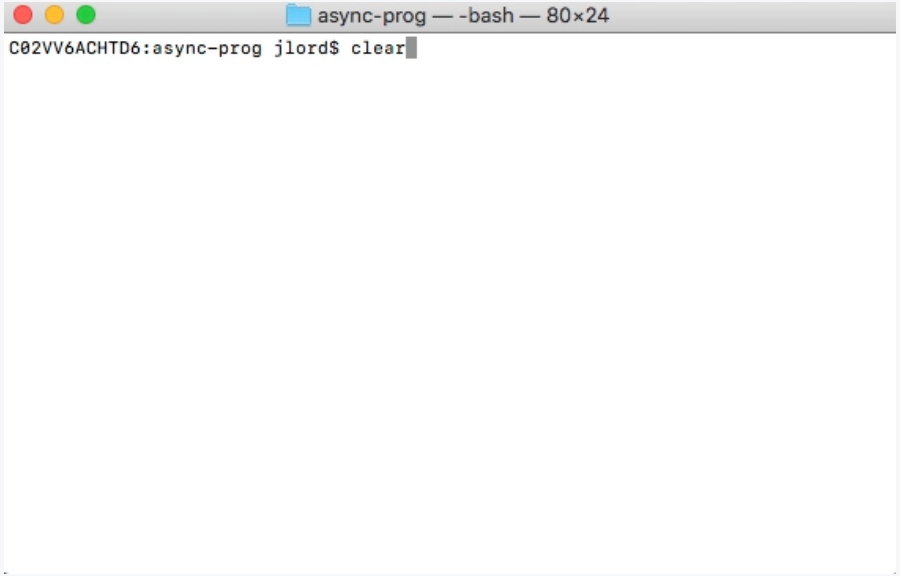
- In node, you can use the event emitter
- Create you own events for a library

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.Events.on("breadFetched", () => {
    PBnJMaker.fetchPeanutButter();
});
PBnJMaker.Events.on("peanutButterFetched", () => PBnJMaker.fetchJam());
PBnJMaker.Events.on("jamFetched", () => PBnJMaker.spreadPeanutButter());
PBnJMaker.Events.on("peanutButterSpreaded", () => PBnJMaker.spreadJam());
PBnJMaker.Events.on("jamSpreaded", () => PBnJMaker.closeSandwich());
PBnJMaker.Events.on("sandwichClosed", (result) => console.log("Eat that " + result));

PBnJMaker.fetchBread();

```

Code

```
let PBnJMaker = require("./pbnjmaker");

PBnJMaker.Events.on("breadFetched", () => {
    PBnJMaker.fetchPeanutButter();
});
PBnJMaker.Events.on("peanutButterFetched", () => PBnJMaker.fetchJam());
PBnJMaker.Events.on("jamFetched", () => PBnJMaker.spreadPeanutButter());
PBnJMaker.Events.on("peanutButterSpreaded", () => PBnJMaker.spreadJam());
PBnJMaker.Events.on("jamSpreaded", () => PBnJMaker.closeSandwich());
PBnJMaker.Events.on("sandwichClosed", (result) => console.log("Eat that " + result));

PBnJMaker.fetchBread();

```

Demo

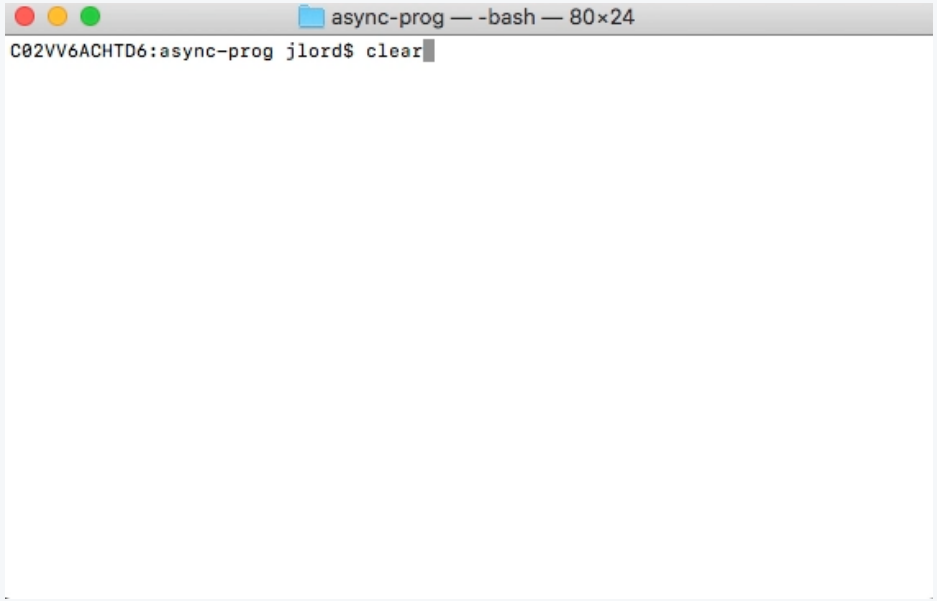


@joel__lord
#confoo

Code

```
let PBnJMaker = require("./pbnjmaker");

let ingredientsFetched = 0;
let ingredientsSpreaded = 0;

let ingredientFetched = () => {
    ingredientsFetched++;
    if (ingredientsFetched === 3) {
        PBnJMaker.Events.emit("ingredientsFetched");
    }
};

let spreaded = () => {
    ingredientsSpreaded++;
    if (ingredientsSpreaded === 2) {
        PBnJMaker.Events.emit("ingredientsSpreaded");
    }
};

PBnJMaker.Events.on("breadFetched", ingredientFetched);
PBnJMaker.Events.on("peanutButterFetched", ingredientFetched);
PBnJMaker.Events.on("jamFetched", ingredientFetched);

PBnJMaker.Events.on("ingredientsFetched", () => {
    PBnJMaker.spreadPeanutButter();
    PBnJMaker.spreadJam();
});

PBnJMaker.Events.on("peanutButterSpreaded", spreaded);
PBnJMaker.Events.on("jamSpreaded", spreaded);

PBnJMaker.Events.on("ingredientsSpreaded", () => {
    PBnJMaker.closeSandwich();
});

PBnJMaker.Events.on("sandwichClosed", (result) => console.log("Eat that " + result));

PBnJMaker.fetchBread();
PBnJMaker.fetchPeanutButter();
PBnJMaker.fetchJam();

```

Demo



@joel__lord
#confoo


Reactive Stream
(2017)

RxJS

- Manipulate data, synchronous or asynchronous as streams
- Manipulate streams like arrays

```
getDataFromLocalMemory()
    .filter (s => s != null)
    .map(s => `${s} transformed`)
    .forEach(s => console.log(`next => ${s}`))
```

```
getDataFromNetwork()
    .filter (s => s != null)
    .map(s => `${s} transformed`)
    .subscribe(s => console.log(`next => ${s}`))
```

Code

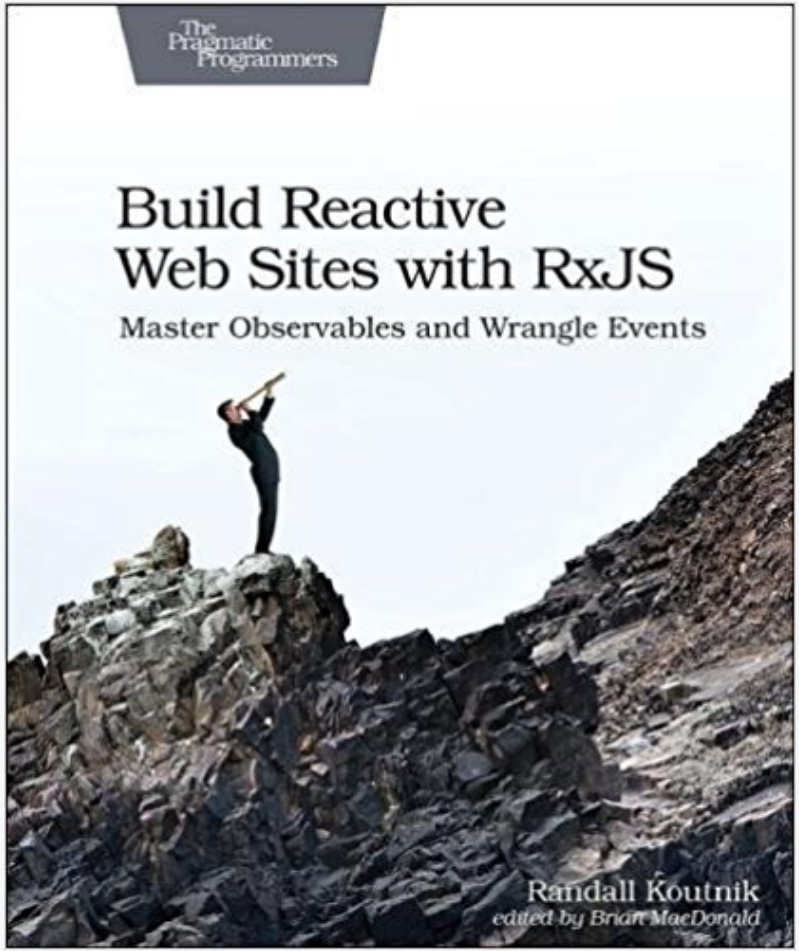
```
1 const Rx = require("rxjs");
2 let PBnJMaker = require("../pbnjmaker");
3
4 const ingredients = ["bread", "peanutButter", "jam"];
5
6 const ingredientEvents = ingredients.map((ingredient) => {
7     return Rx.Observable.fromEvent(PBnJMaker.Events, `${ingredient}Fetched`);
8 });
9
10 const ingredientEvents$ = Rx.Observable.merge(...ingredientEvents);
11
12 const subscription = ingredientEvents$.subscribe(() => {
13     console.log("Just fetched an ingredient");
14 });
```

Demo

@joel__lord
#confoo

RxJS

- RTFM


The Pragmatic Programmers
Build Reactive Web Sites with RxJS
Master Observables and Wrangle Events
Randall Koutnik
edited by Brian MacDonald


In Conclusion
(c’est presque la faim)

Summary

Asynchronous patterns

- The Event Loop
- Callbacks
- Promises
- Generators
- Async/Await
- Events
- Reactive Stream

```
getData(a ⇒ {
    getMoreData(a, b ⇒ {
        getMoreData(b, c ⇒ {
            getMoreData(c, d ⇒ {
                getMoreData(d, e ⇒ {
                    console.log(e);
                });
            });
        });
    });
});
```

Summary

Asynchronous patterns

- The Event Loop
- Callbacks
- Promises
- Generators
- Async/Await
- Events
- Reactive Stream
- …. How to make a PBnJ sandwich!

```
getData(a ⇒ {
    getMoreData(a, b ⇒ {
        getMoreData(b, c ⇒ {
            getMoreData(c, d ⇒ {
                getMoreData(d, e ⇒ {
                    console.log(e);
                });
            });
        });
    });
});
```

Asynchronicity: concurrency. A tale of

Confoo - Montreal, Canada
March 7, 2018

@joel__lord

joellord