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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18

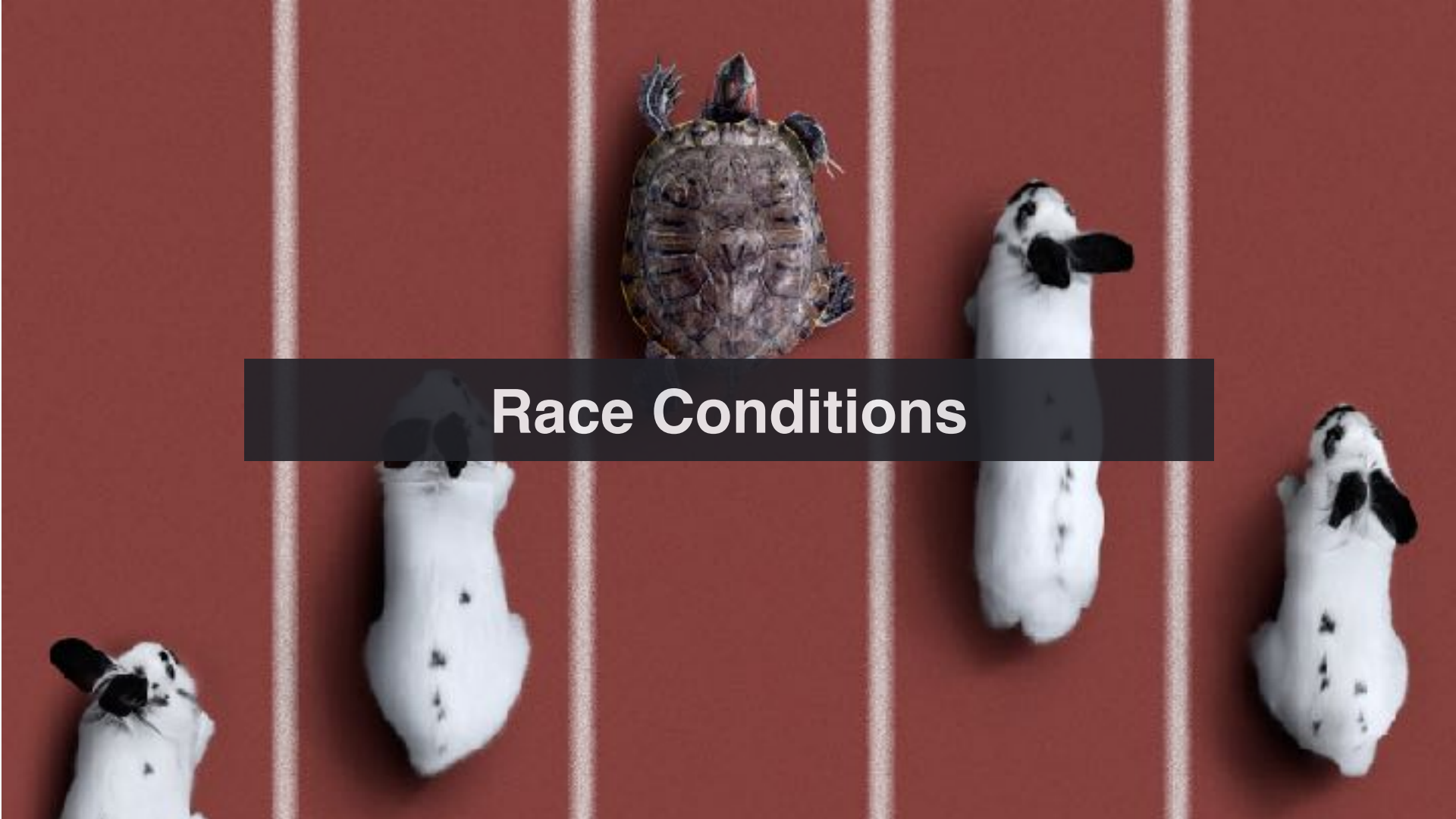
PHP

```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");
});

console.log("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");
req.send(null);

</script>
```

@joel__lord
#iJS18


Race Conditions

About Me

@joel__lord

joellord

WARNING

WARNING

175 slides

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
#iJS18


The Event Loop

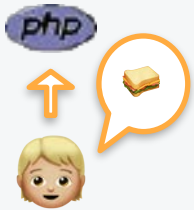
Classic Architecture (PHP, Java)

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)

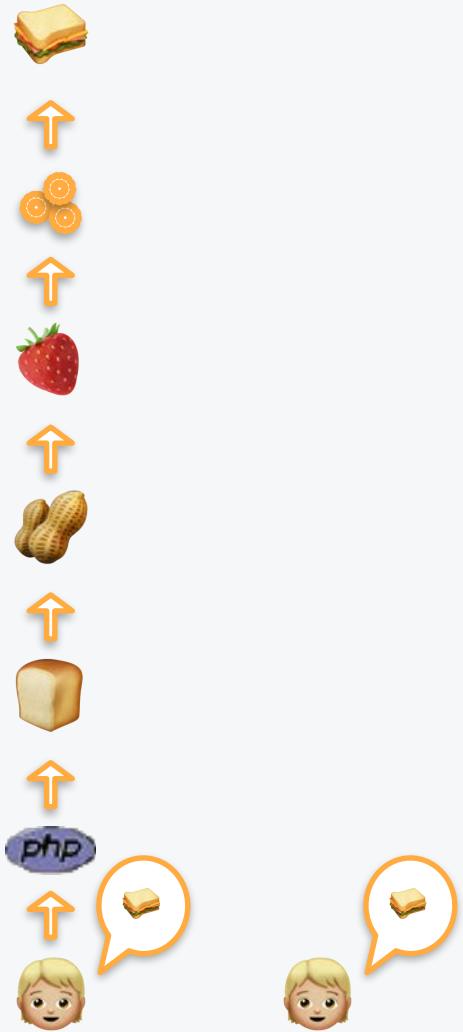
- I want a sandwich!
- Process start
  - Fetches the ingredients
  - Prepares the sandwich
  -

Classic Architecture (PHP, Java)

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

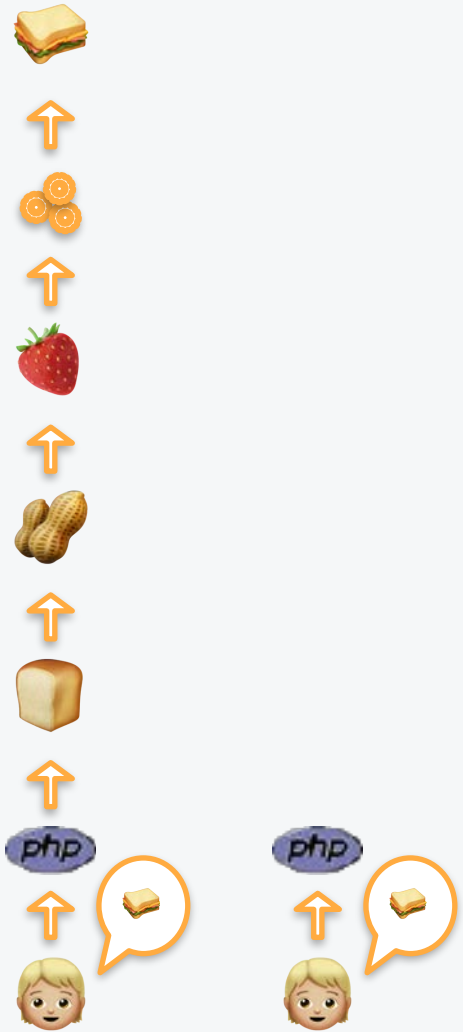
Classic Architecture (PHP, Java)

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



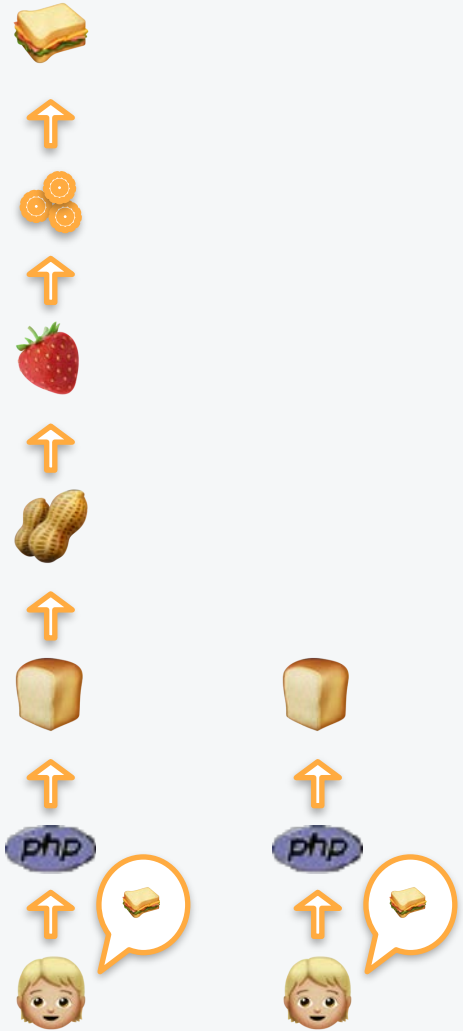
Classic Architecture (PHP, Java)

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



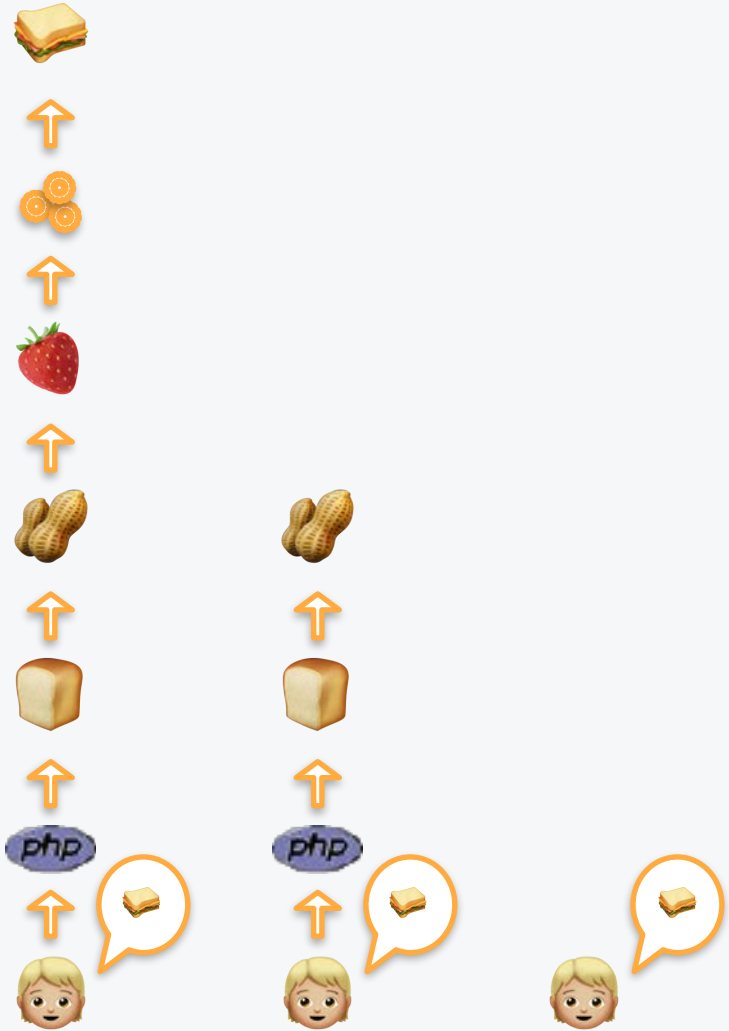
Classic Architecture (PHP, Java)

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



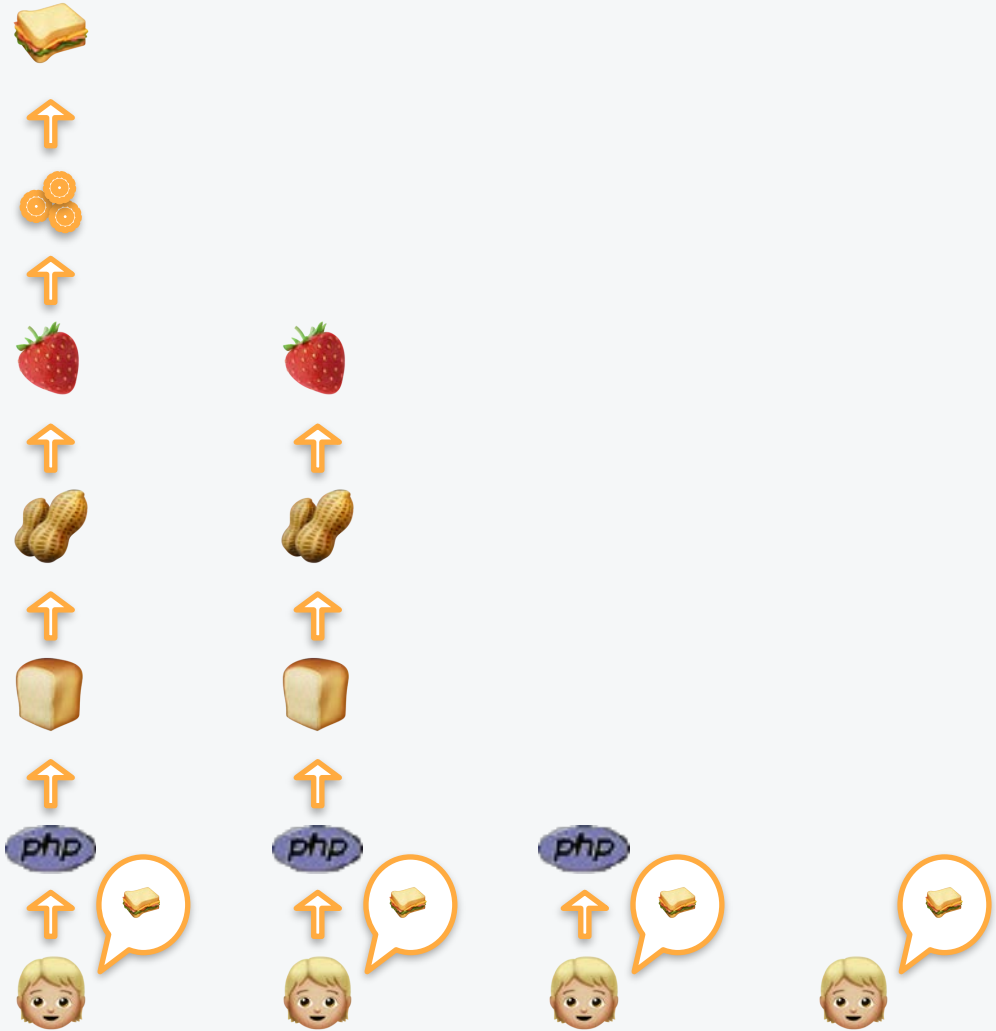
Classic Architecture (PHP, Java)

- 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
  -



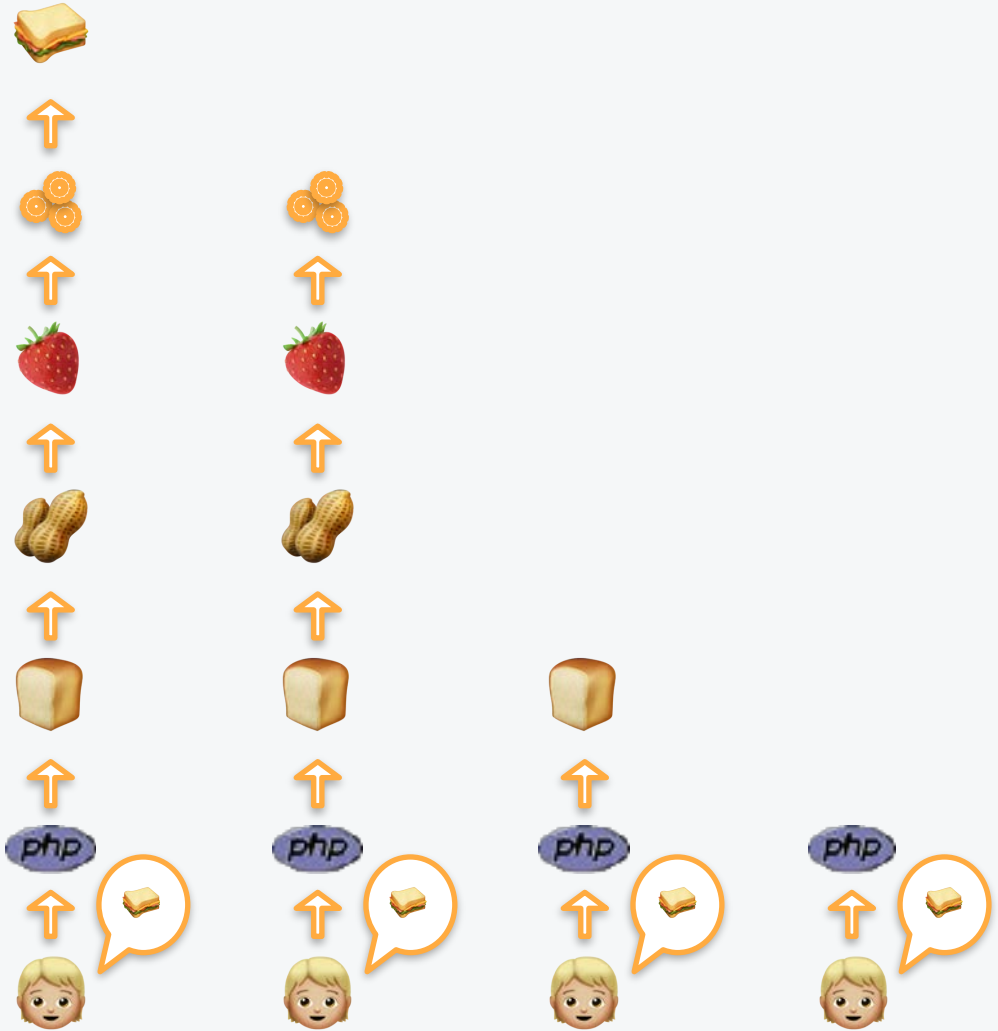
Classic Architecture (PHP, Java)

- 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



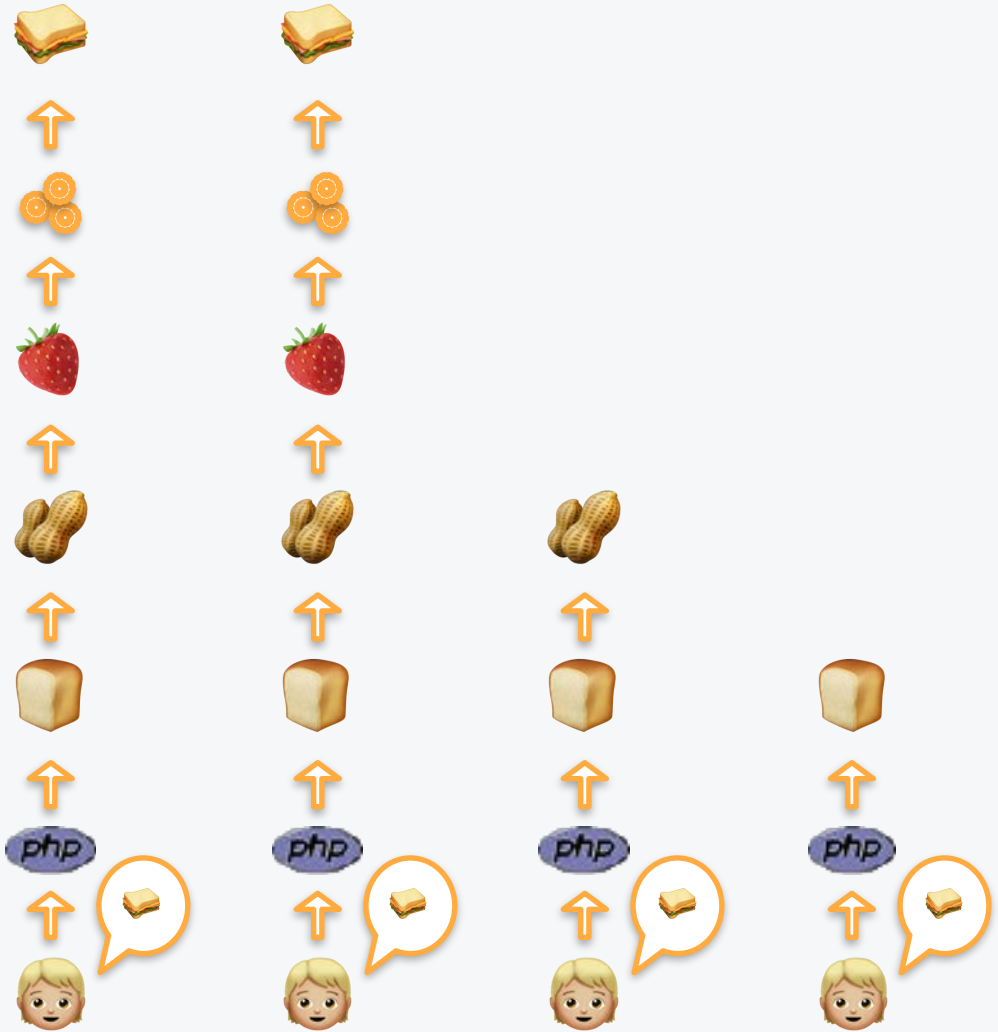
Classic Architecture (PHP, Java)

- 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



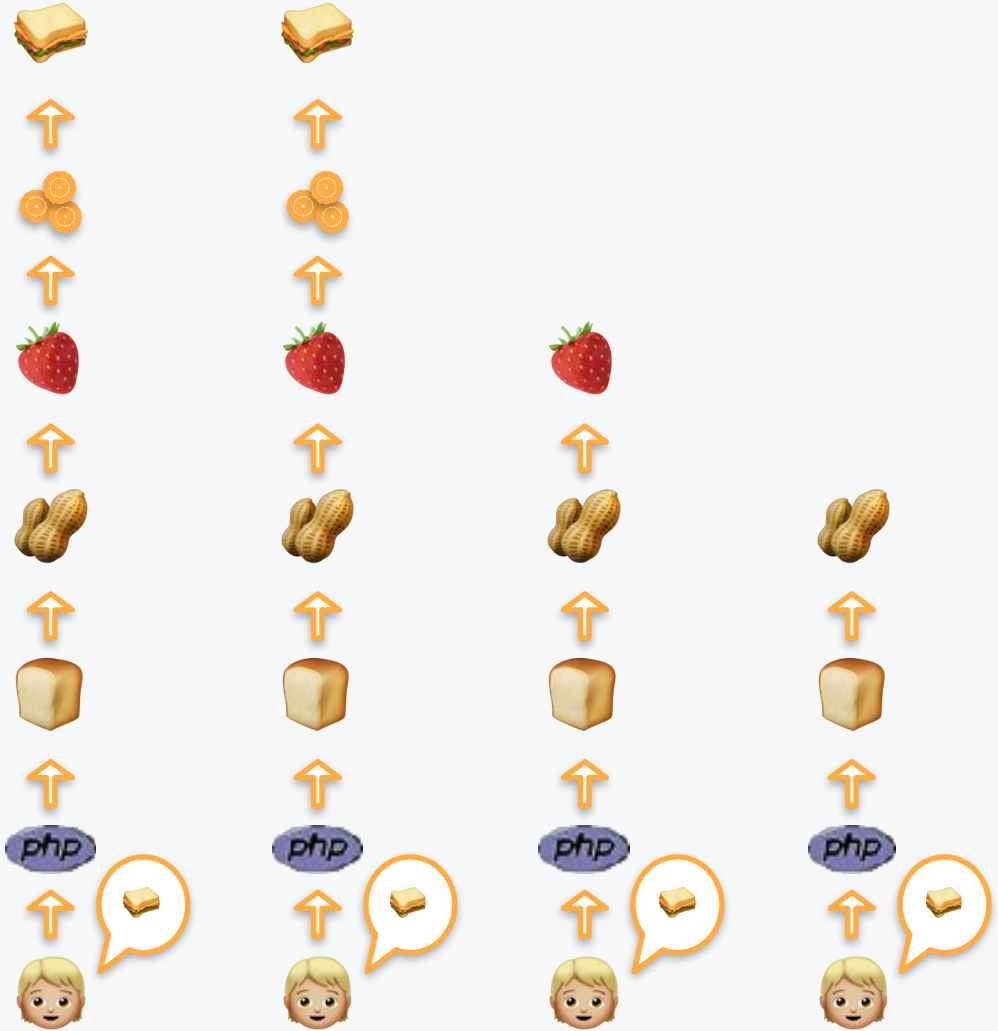
Classic Architecture (PHP, Java)

- 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



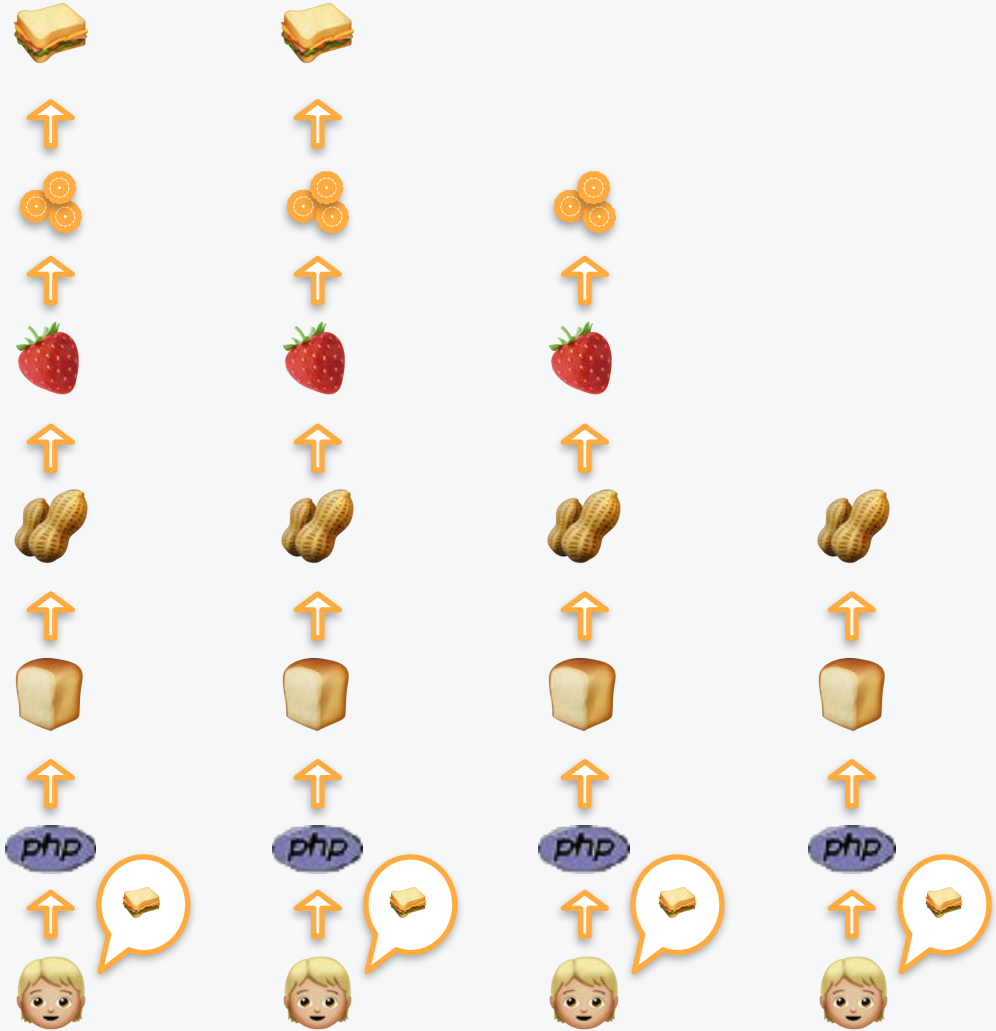
Classic Architecture (PHP, Java)

- 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)

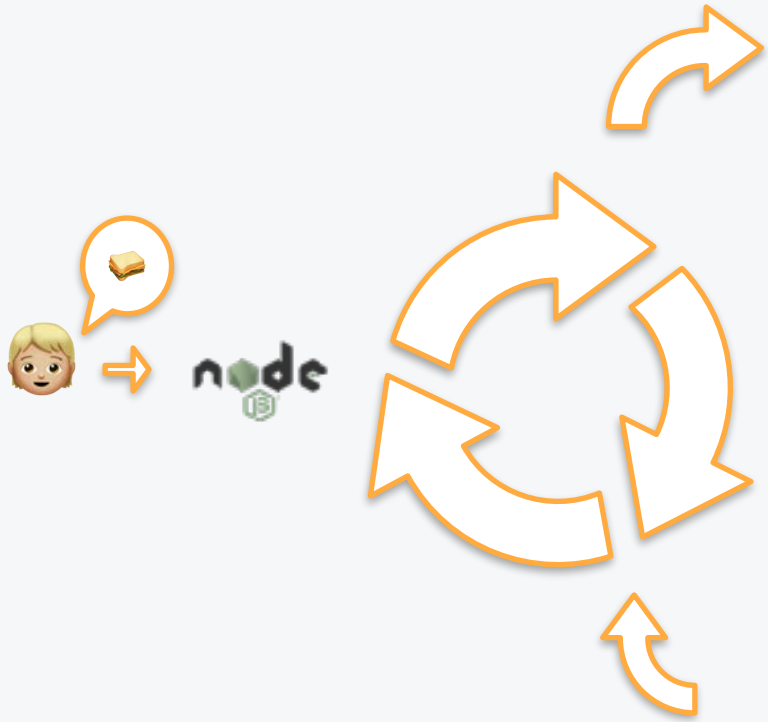
- I want a sandwich!
- Process start
  - Fetches the ingredients
- A new customer arrives: I want a sandwich!
- Another process starts
  - Fetches the ingredients
  - Prepares the sandwich
  - Here's your sandwich

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

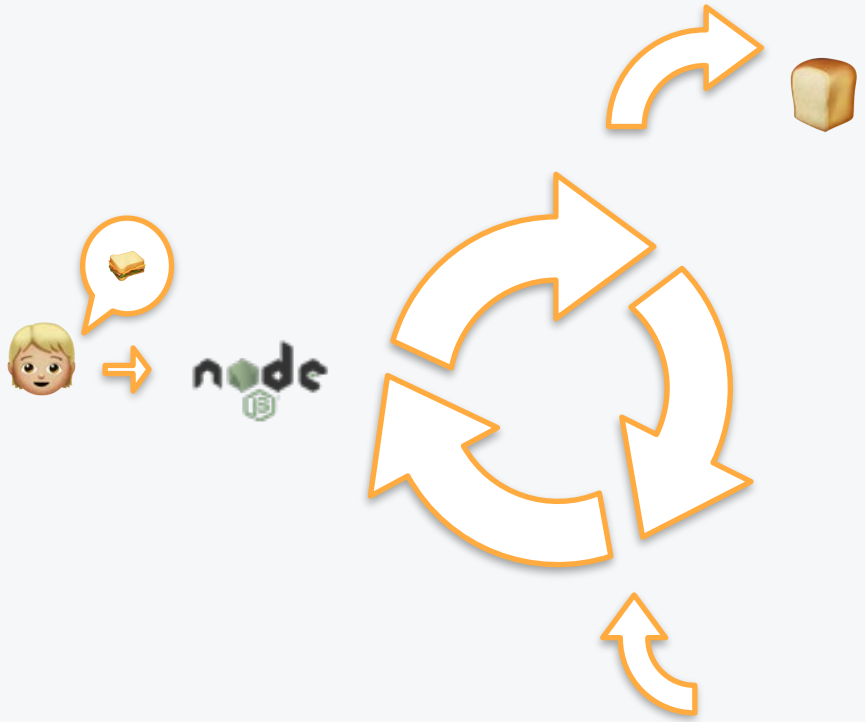
Event Loop (NodeJS)

- 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



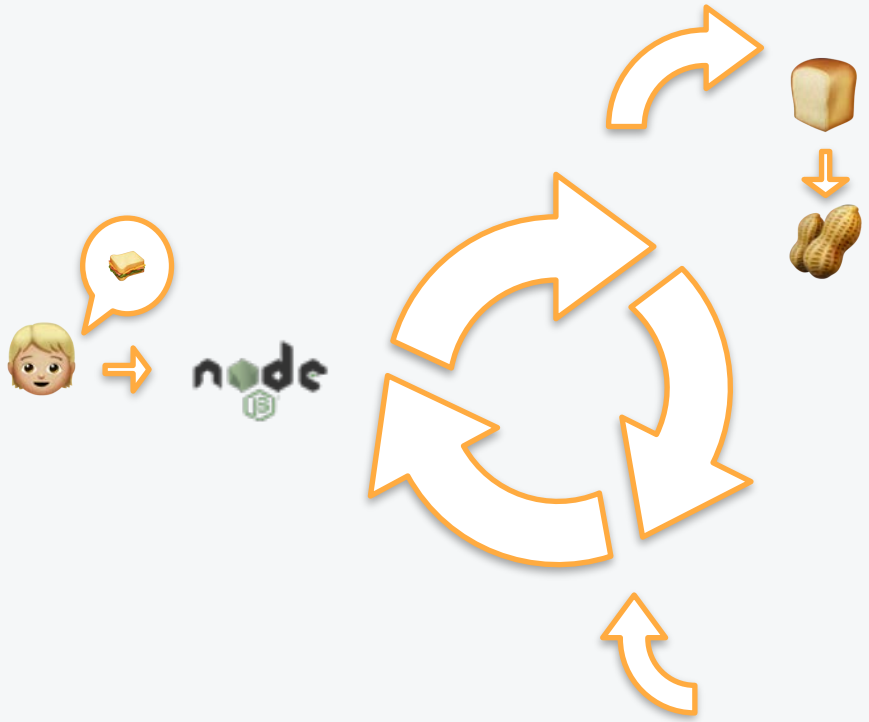
Event Loop (NodeJS)

- 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



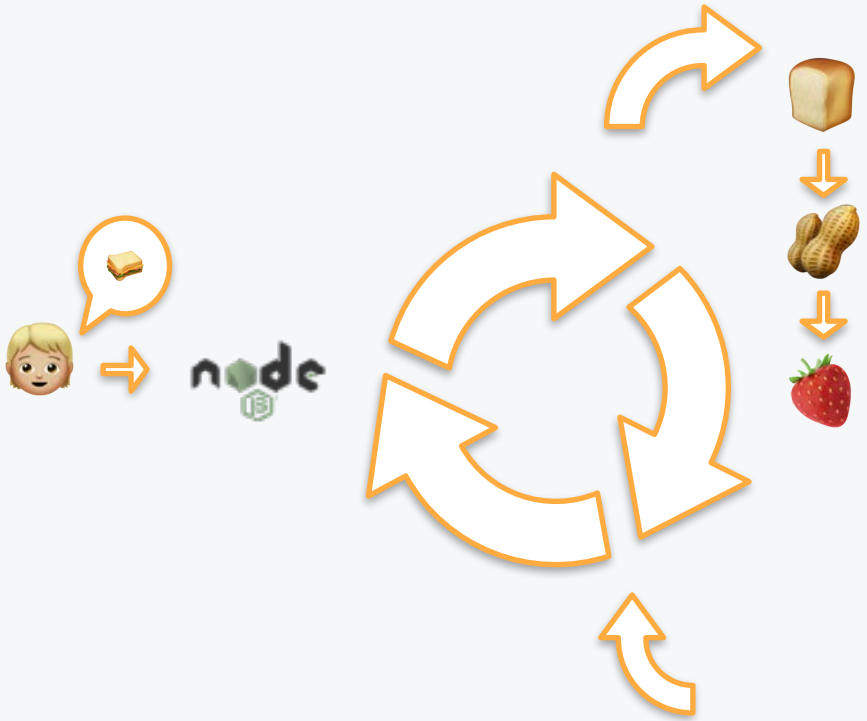
Event Loop (NodeJS)

- 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



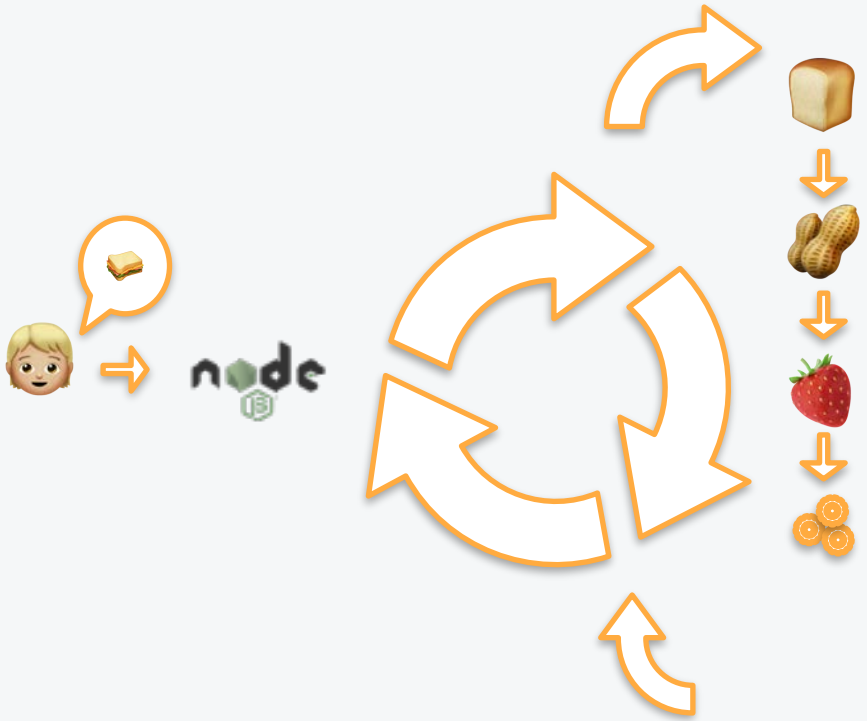
Event Loop (NodeJS)

- 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



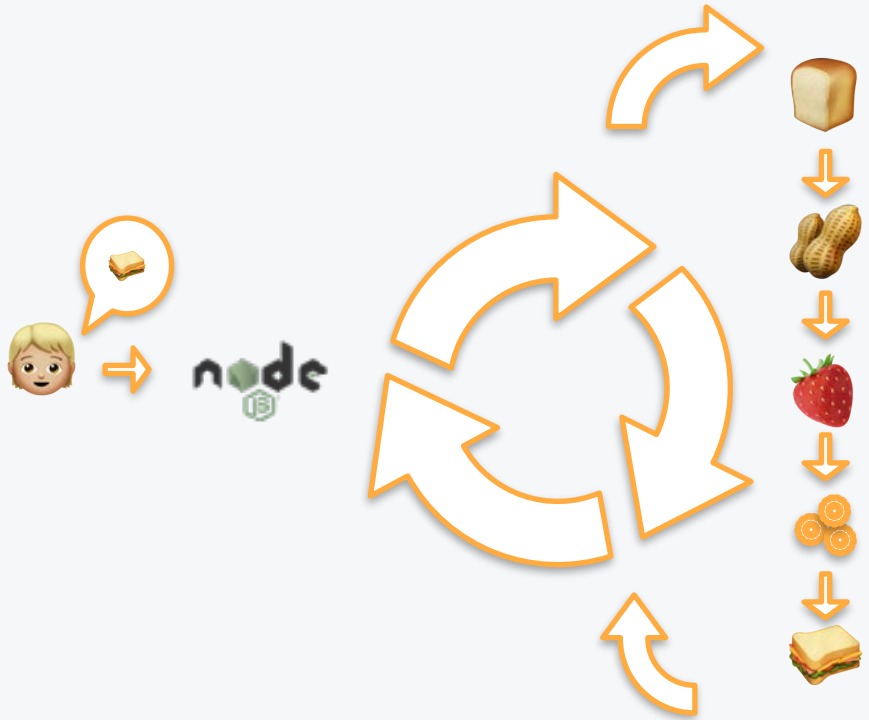
Event Loop (NodeJS)

- 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



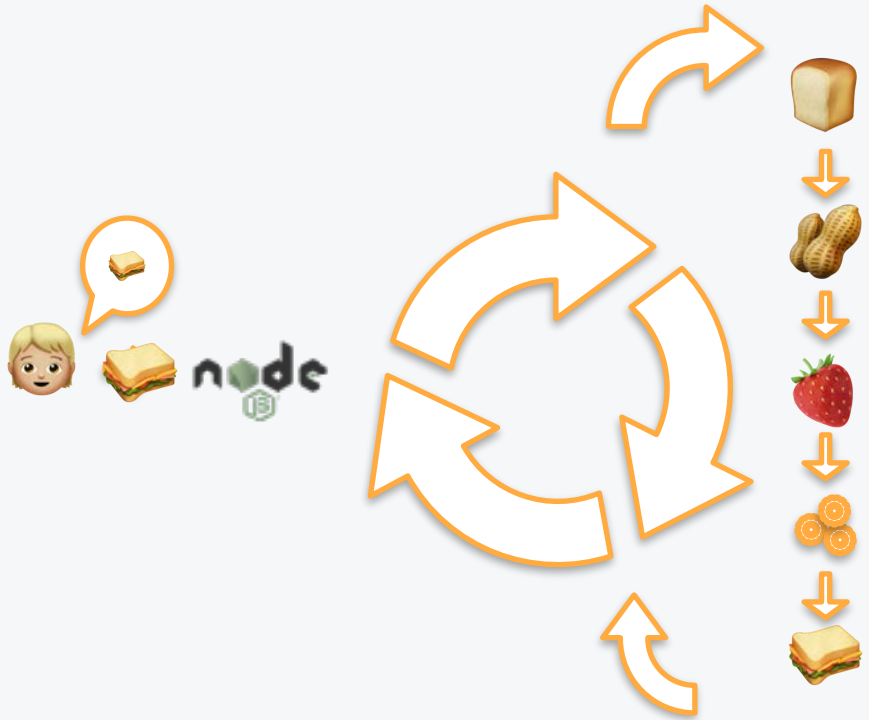
Event Loop (NodeJS)

- 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



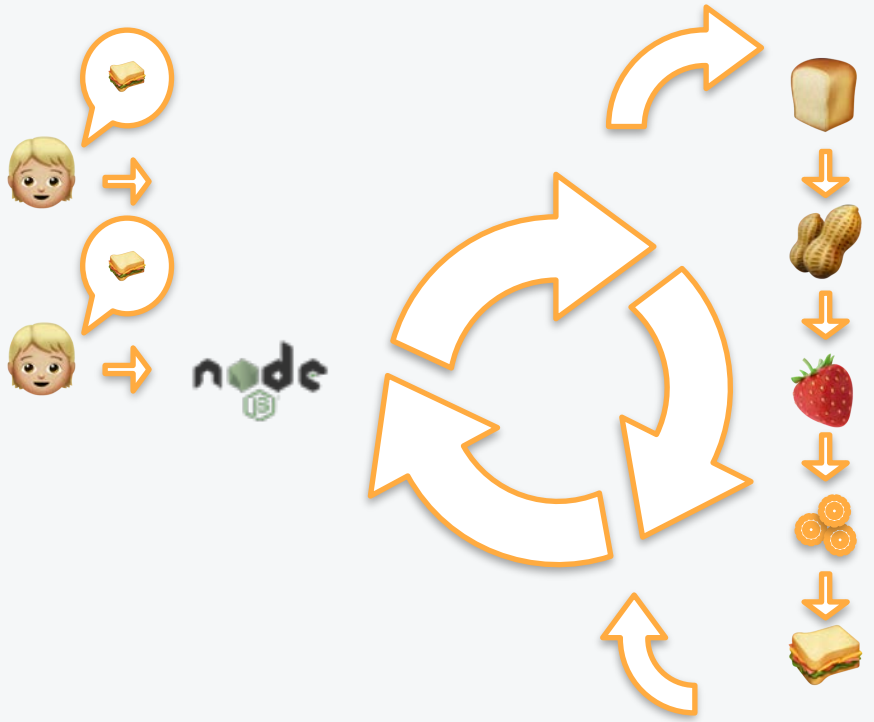
Event Loop (NodeJS)

- 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



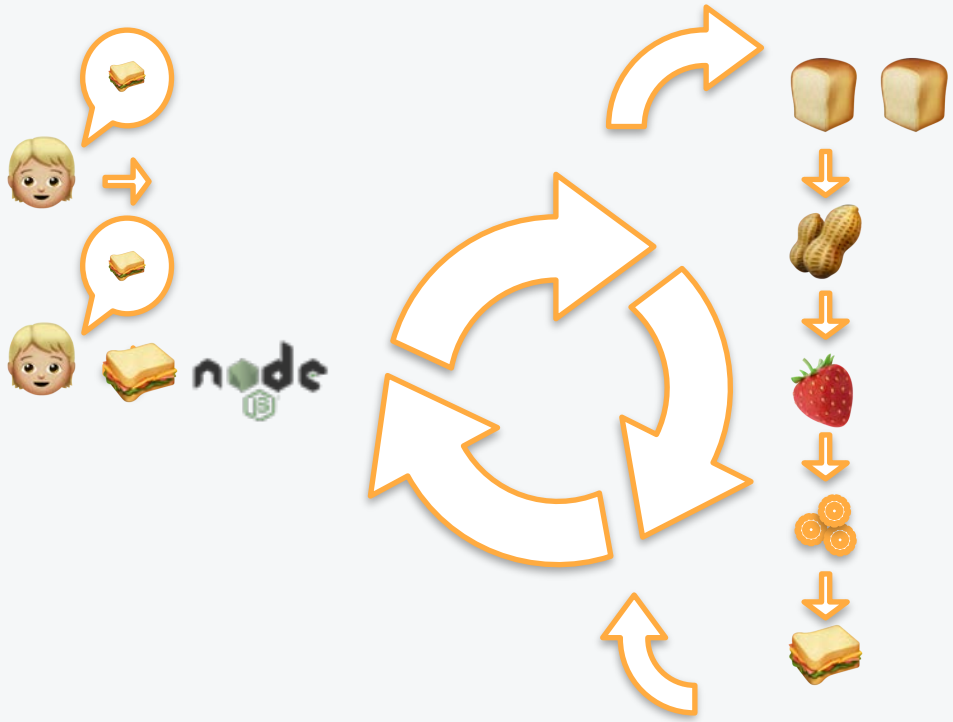
Event Loop (NodeJS)

- 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



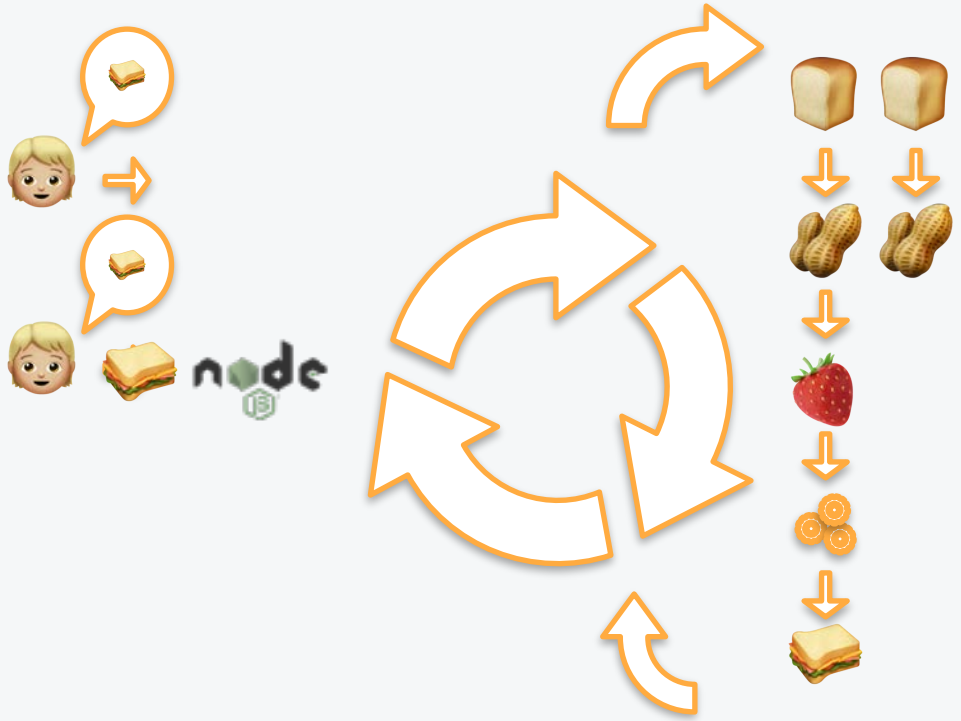
Event Loop (NodeJS)

- 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



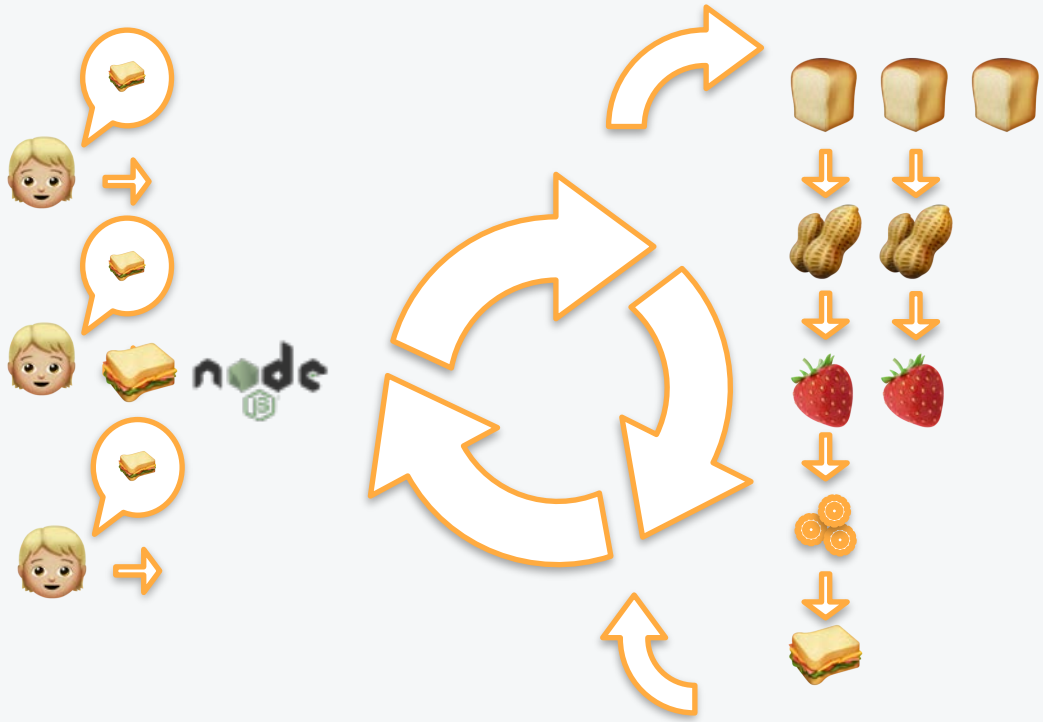
Event Loop (NodeJS)

- 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



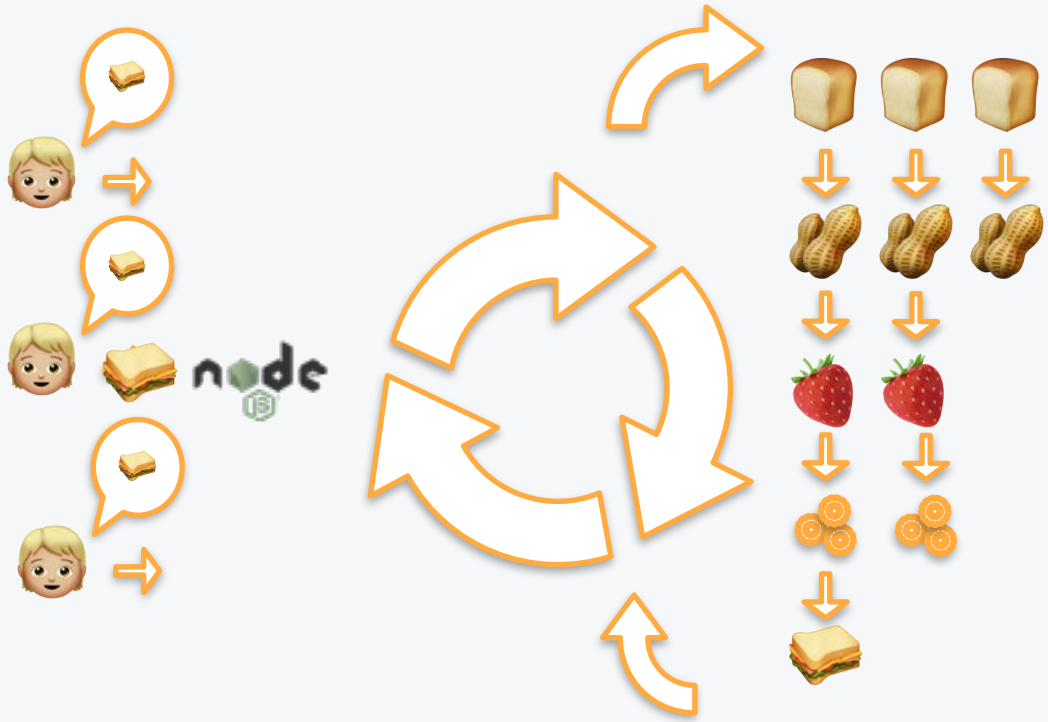
Event Loop (NodeJS)

- 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



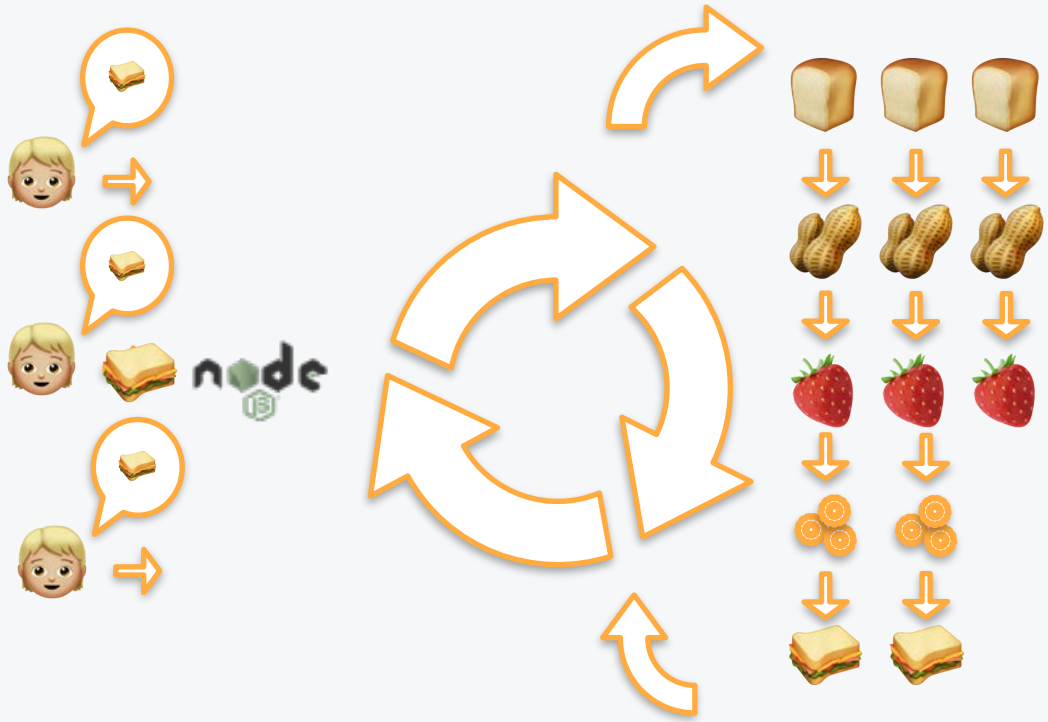
Event Loop (NodeJS)

- 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



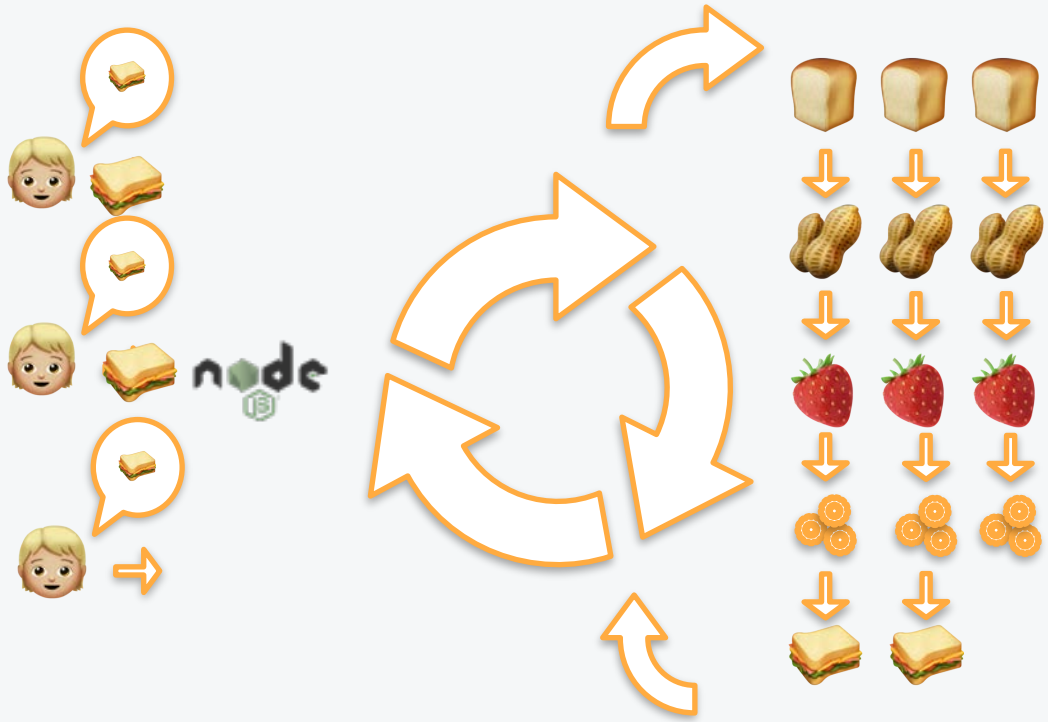
Event Loop (NodeJS)

- 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



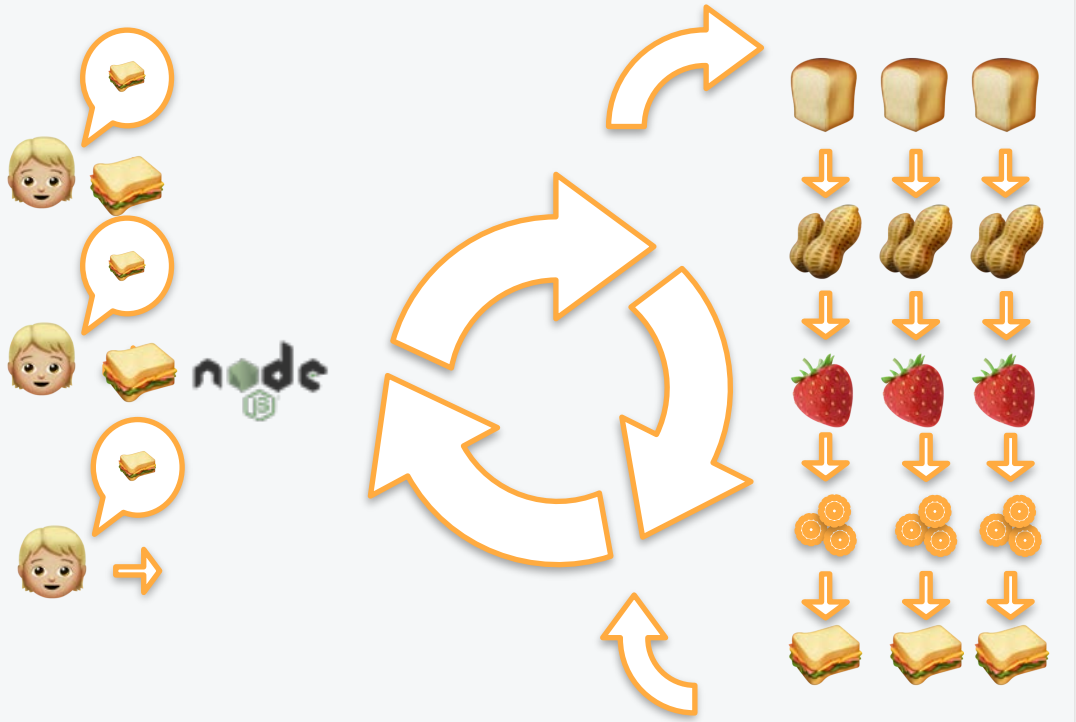
Event Loop (NodeJS)

- 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



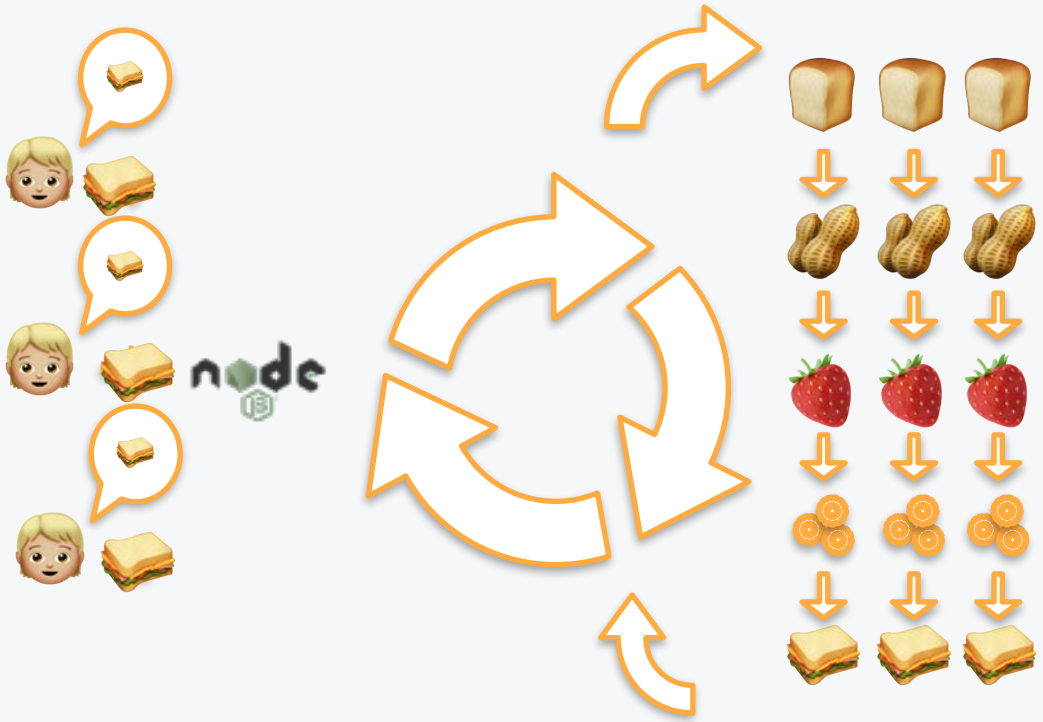
Event Loop (NodeJS)

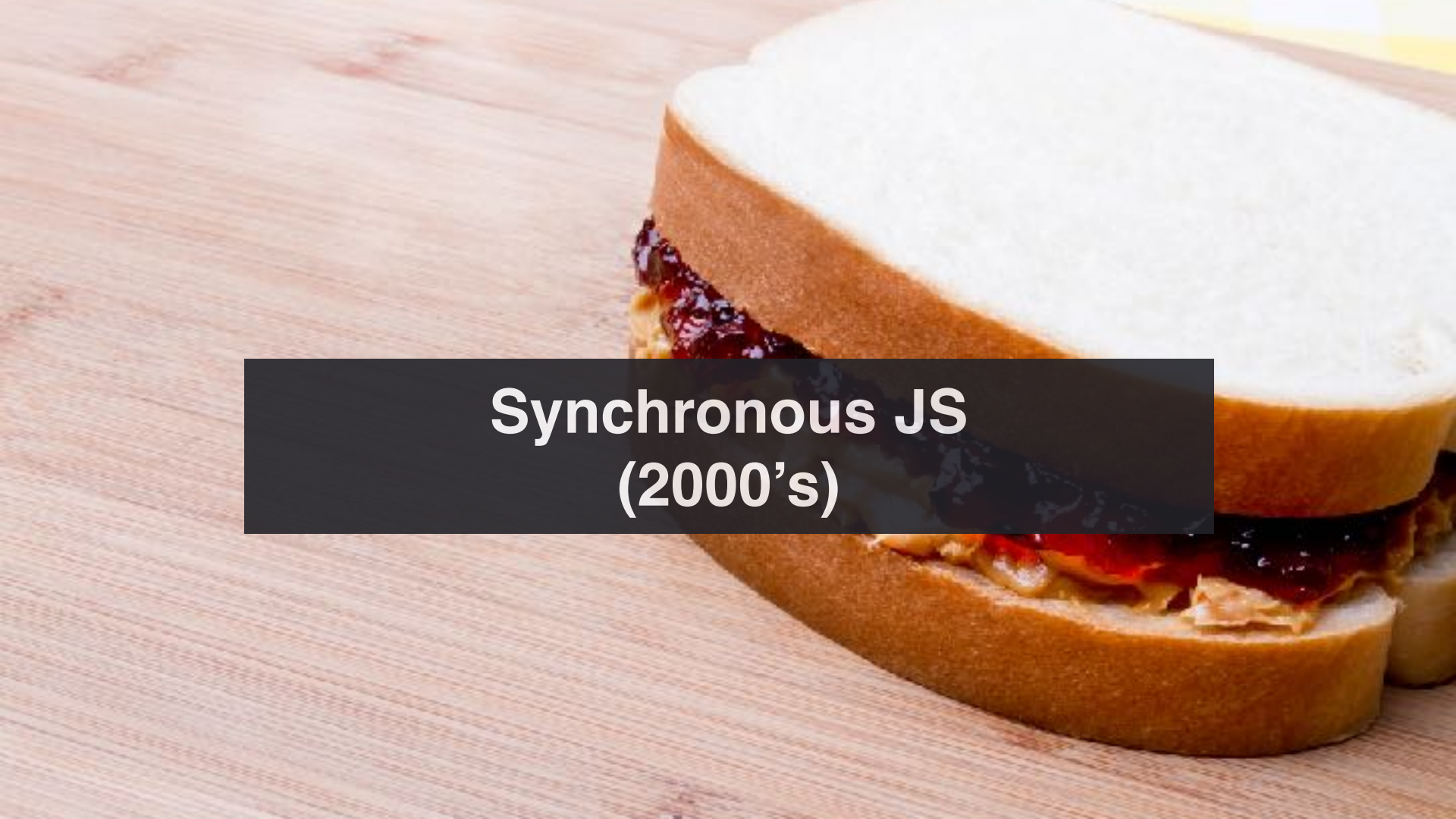
- I want a sandwich!
- Sure
  - “Worker, Make a sandwich”
- A new customer arrives: I want a sandwich!
- Sure
  - “Other worker, make a sandwich”
- Here is your sandwich customer 1



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

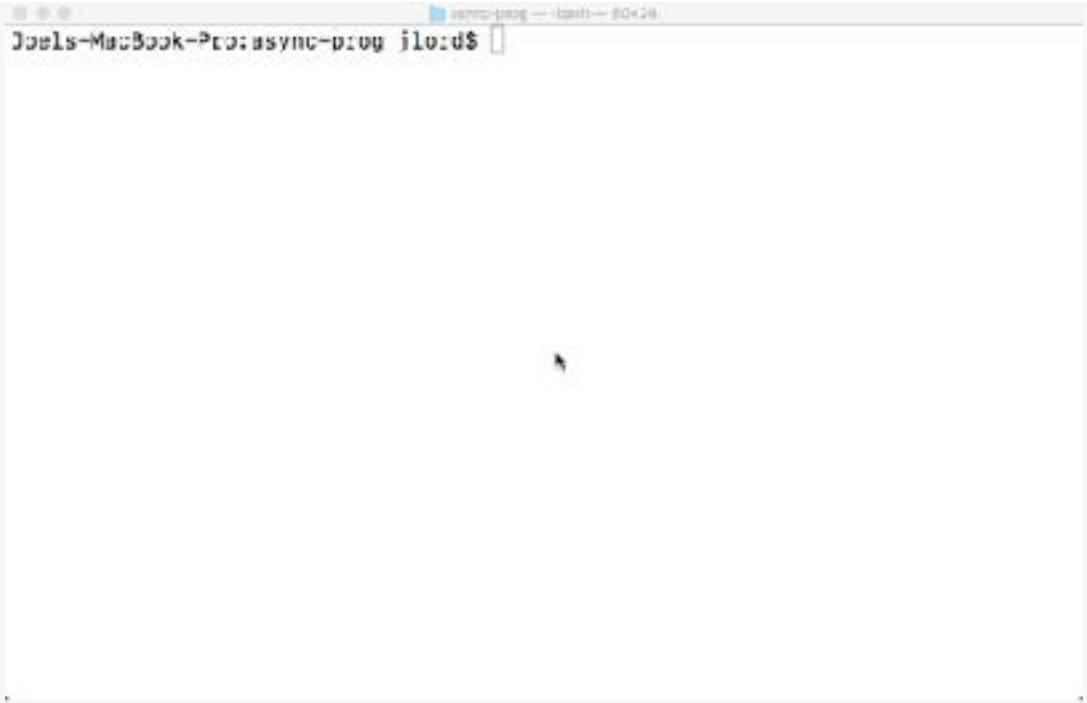


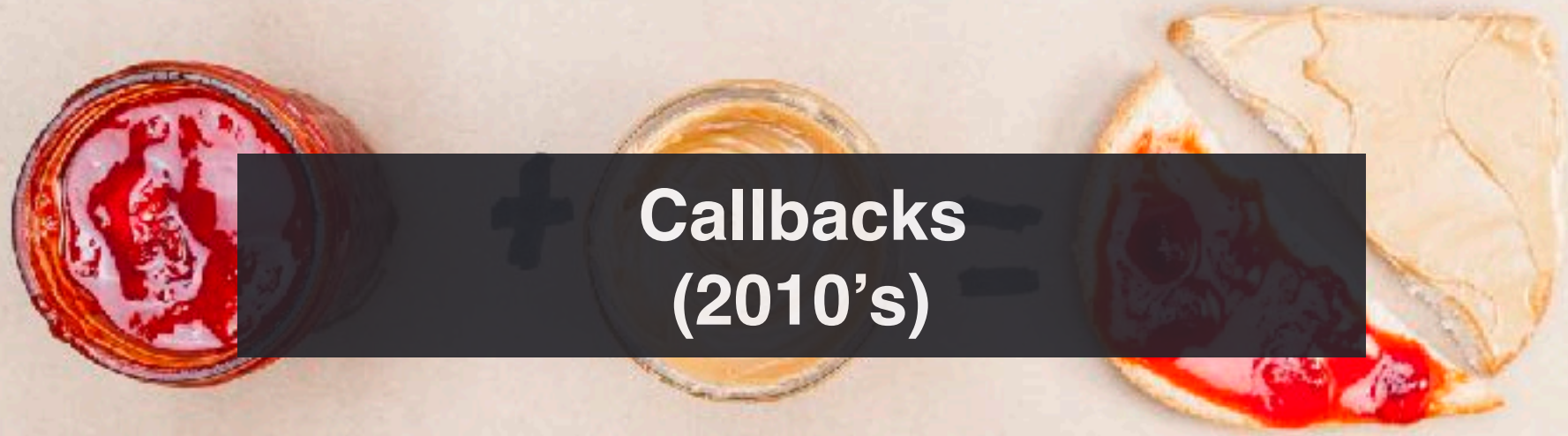

Synchronous JS
(2000’s)

Code

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

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

Result





Callbacks (2010's)

What is a callback

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

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

What is a callback

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

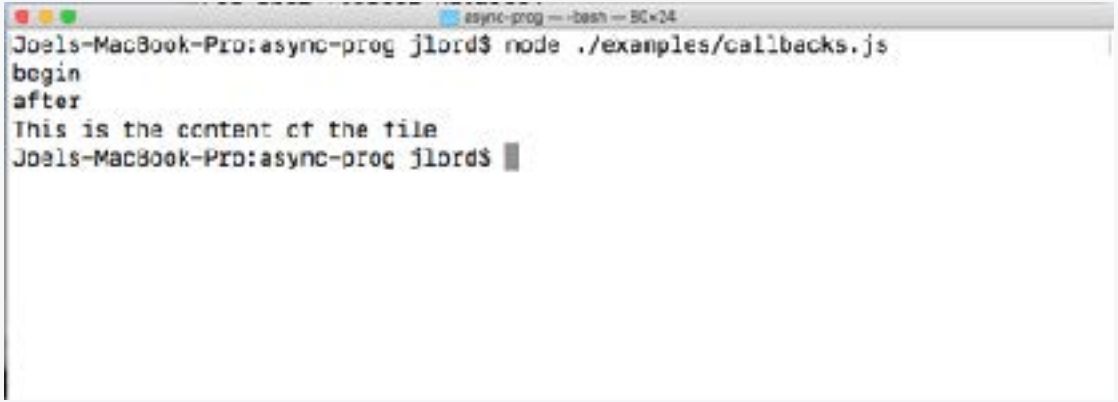
What is a callback

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

```
//Asynchronous
console.log("begin");
fs.readFile("./cb.txt", (err, data) => {
    console.log(“File content”);
});
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

```
//Asynchronous
console.log("begin");
fs.readFile("./cb.txt", (err, data) => {
    console.log(“File content”);
});
console.log("after");
```

What is a callback

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

```
//Asynchronous
.....................
..................... (err, data) => {
   console.log(“File content”);
}..
.....................
```

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");
```

```
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
```

What is a callback

Asynchronous callback and the event loop

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

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

```
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
```

What is a callback

Asynchronous callback and the event loop

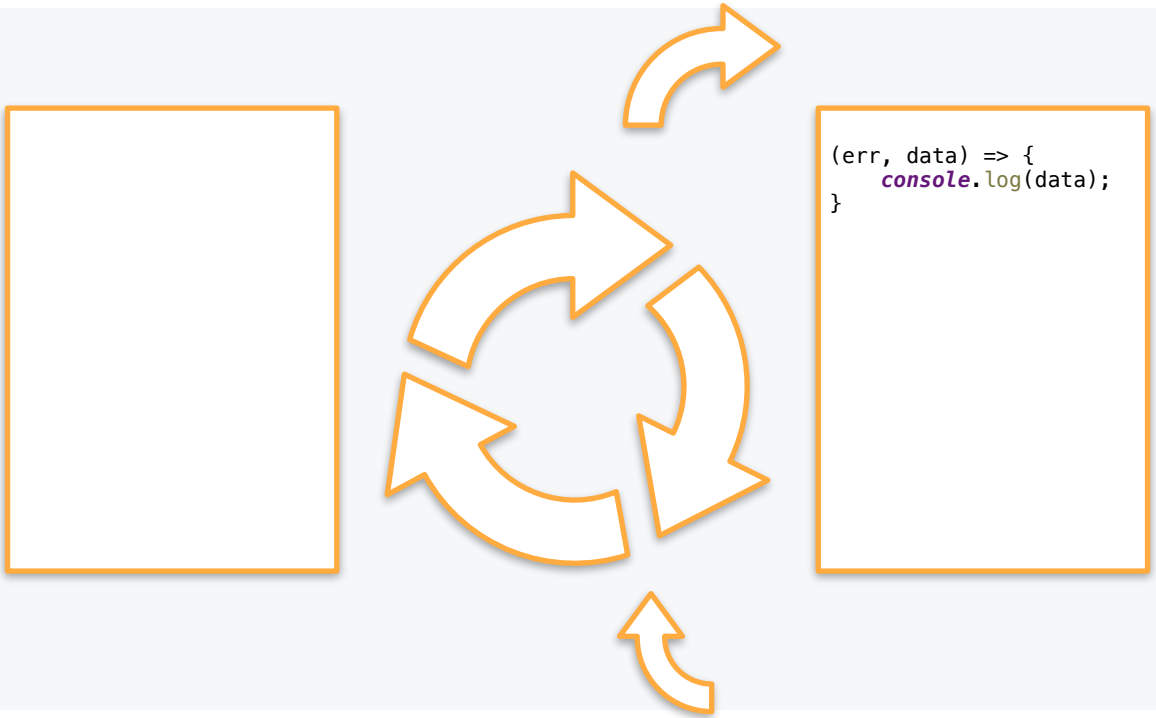
```
console.log("after");
```

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

```
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
```

What is a callback

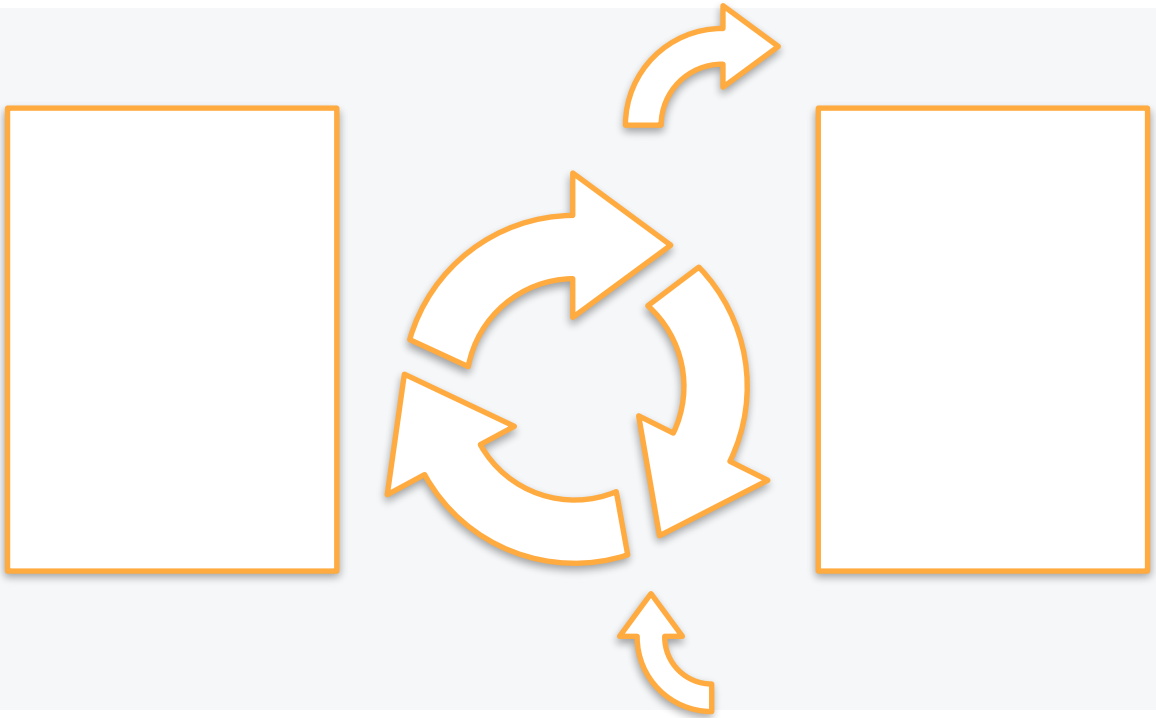
Asynchronous callback and the event loop

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

```
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
after
```

What is a callback

Asynchronous callback and the event loop



```
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);
                    });
                });
            });
        });
    });
});
```

Result

Callbacks

- It works!
- but…

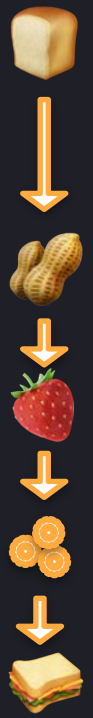
```
let PBnJMaker = require("../pbnjmaker");

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… the sandwich");
                            } else {
                                PBnJMaker.spreadJam((err) => {
                                    if (err) {
                                        console.log("An error occurred … the sandwich");
                                    } else {
                                        PBnJMaker.closeSandwich((err, result) => {
                                            if (err) {
                                                console.log("An error occurred");
                                            } else {
                                                console.log("Eat that " + result);
                                            }
                                        });
                                    }
                                });
                            }
                        });
                    }
                });
            }
        });
    }
});
```

Callback Hell

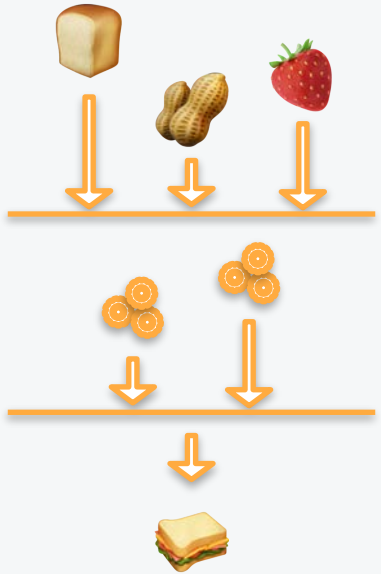
```
let PBnJMaker = require("../pbnjmaker");

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… ingredients");
            } else {
                PBnJMaker.fetchJam((err) => {
```

Parallel events

Step by step



Parallel events



@joel__lord
#iJS18

Code

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

@joel__lord
#iJS18

Code

```
let ingredients = 0;
let steps = 0;
```

@joel__lord
#iJS18

Code

```
function fetchIngredients() {

}
```

@joel__lord
#iJS18

Code

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

@joel__lord
#iJS18

Code

```
function handleIngredients(err) {
    if (err) return console.log("Error");
    ingredients++;
    if (ingredients === 3) {
        prepareSandwich();
    }
}
```

@joel__lord
#iJS18

Code

```
function prepareSandwich() {

PBnJMaker.spreadPeanutButter(preparationStepCompleted);
   PBnJMaker.spreadJam(preparationStepCompleted);
}
```

@joel__lord
#iJS18

Code

```
function preparationStepCompleted(err) {
    if (err) return console.log("Error");
    steps++;
    if (steps === 2) {
        completeSandwich();
    }
}
```

@joel__lord
#iJS18

Code

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

@joel__lord
#iJS18

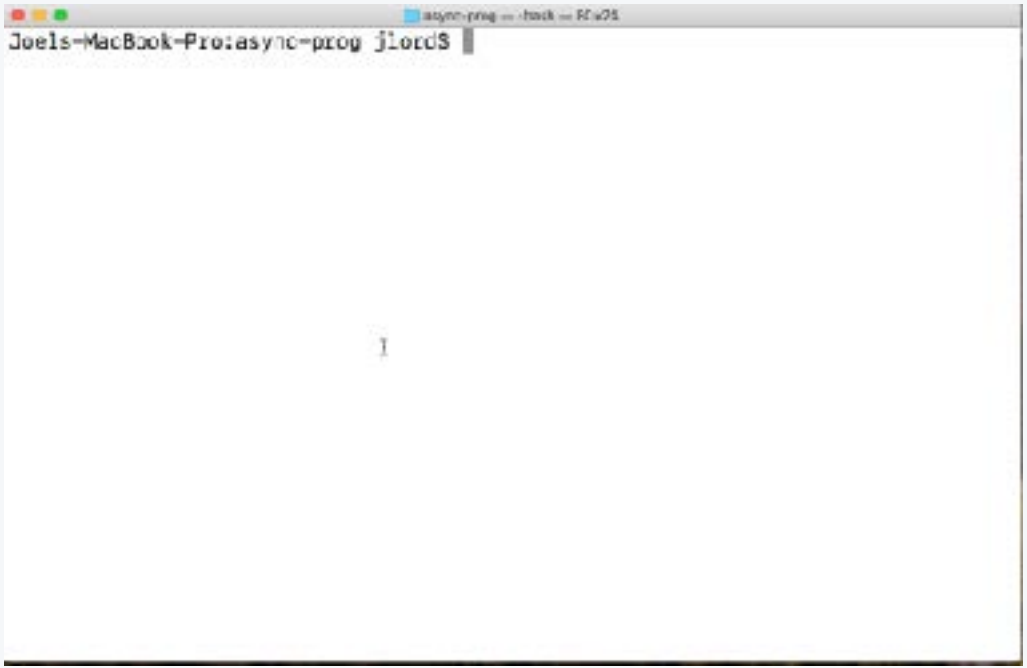
Code

```
fetchIngredients();
```

@joel__lord
#iJS18

Result

Parallel




Promises
(2014)

What is a promise?

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

```
//Asynchronous
console.log("begin");
asyncPromise().then(onSuccess, onFailure);
console.log("after");
```

What is a promise?

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

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

What is a promise?

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

```
//Asynchronous
console.log("begin");
fs.readFile("./cb.txt")
  .then((data) => {
    console.log(data);
  }).catch((err) => {
    console.log(err);
  });
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

```
PBnJMaker.fetchBread()
```

@joel__lord
#iJS18

Code

```
1 let PBnJMaker = require("./pbnjmaker");
2
3 PBnJMaker.fetchBread().then(() => {
4     return PBnJMaker.fetchPeanutButter();
5 }).then(() => {
6     return PBnJMaker.fetchJam();
7 }).then(() => {
8     return PBnJMaker.spreadPeanutButter();
9 }).then(() => {
10    return PBnJMaker.spreadJam();
11 }).then(() => {
12    return PBnJMaker.closeSandwich();
13 }).then((result) => {
14    console.log("Eat that " + result);
15 });
```

Result

```
PBnJMaker.fetchBread().then(() => {
```

@joel__lord
#iJS18

Code

```
1 let PBnJMaker = require("./pbnjmaker");
2
3 PBnJMaker.fetchBread().then(() => {
4     return PBnJMaker.fetchPeanutButter();
5 }).then(() => {
6     return PBnJMaker.fetchJam();
7 }).then(() => {
8     return PBnJMaker.spreadPeanutButter();
9 }).then(() => {
10    return PBnJMaker.spreadJam();
11 }).then(() => {
12    return PBnJMaker.closeSandwich();
13 }).then((result) => {
14    console.log("Eat that " + result);
15 });
```

Result

```
PBnJMaker.fetchBread().then(() => {
```

@joel__lord
#iJS18

Code

```
1 let PBnJMaker = require("./pbnjmaker");
2
3 PBnJMaker.fetchBread().then(() => {
4     return PBnJMaker.fetchPeanutButter();
5 }).then(() => {
6     return PBnJMaker.fetchJam();
7 }).then(() => {
8     return PBnJMaker.spreadPeanutButter();
9 }).then(() => {
10    return PBnJMaker.spreadJam();
11 }).then(() => {
12    return PBnJMaker.closeSandwich();
13 }).then((result) => {
14    console.log("Eat that " + result);
15 });
```

Result

```
PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
})
```

@joel__lord
#iJS18

Code

```
1 let PBnJMaker = require("./pbnjmaker");
2
3 PBnJMaker.fetchBread().then(() => {
4     return PBnJMaker.fetchPeanutButter();
5 }).then(() => {
6     return PBnJMaker.fetchJam();
7 }).then(() => {
8     return PBnJMaker.spreadPeanutButter();
9 }).then(() => {
10    return PBnJMaker.spreadJam();
11 }).then(() => {
12    return PBnJMaker.closeSandwich();
13 }).then((result) => {
14    console.log("Eat that " + result);
15 });
```

Result

```
PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
```

@joel__lord
#iJS18

Code

```
1 let PBnJMaker = require("./pbnjmaker");
2
3 PBnJMaker.fetchBread().then(() => {
4     return PBnJMaker.fetchPeanutButter();
5 }).then(() => {
6     return PBnJMaker.fetchJam();
7 }).then(() => {
8     return PBnJMaker.spreadPeanutButter();
9 }).then(() => {
10    return PBnJMaker.spreadJam();
11 }).then(() => {
12    return PBnJMaker.closeSandwich();
13 }).then((result) => {
14    console.log("Eat that " + result);
15 });
```

Result

```
PBnJMaker.fetchBread().then(() => {
    return PBnJMaker.fetchPeanutButter();
}).then(() => {
    return PBnJMaker.fetchJam();
```

@joel__lord
#iJS18

Code

```
1  let PBnJMaker = require("./pbnjmaker");
2
3  PBnJMaker.fetchBread().then(() => {
4      return PBnJMaker.fetchPeanutButter();
5  }).then(() => {
6      return PBnJMaker.fetchJam();
7  }).then(() => {
8      return PBnJMaker.spreadPeanutButter();
9  }).then(() => {
10     return PBnJMaker.spreadJam();
11 }).then(() => {
12     return PBnJMaker.closeSandwich();
13 }).then((result) => {
14     console.log("Eat that " + result);
15 });
```

Result

```
}).then(() => {
    return PBnJMaker.spreadPeanutButter();
```

@joel__lord
#iJS18

Code

```
1  let PBnJMaker = require("./pbnjmaker");
2
3  PBnJMaker.fetchBread().then(() => {
4      return PBnJMaker.fetchPeanutButter();
5  }).then(() => {
6      return PBnJMaker.fetchJam();
7  }).then(() => {
8      return PBnJMaker.spreadPeanutButter();
9  }).then(() => {
10     return PBnJMaker.spreadJam();
11 }).then(() => {
12     return PBnJMaker.closeSandwich();
13 }).then((result) => {
14     console.log("Eat that " + result);
15 });
```

Result

```
}).then(() => {
    return PBnJMaker.spreadJam();
```

@joel__lord
#iJS18

Code

```
1 let PBnJMaker = require("./pbnjmaker");
2
3 PBnJMaker.fetchBread().then(() => {
4     return PBnJMaker.fetchPeanutButter();
5 }).then(() => {
6     return PBnJMaker.fetchJam();
7 }).then(() => {
8     return PBnJMaker.spreadPeanutButter();
9 }).then(() => {
10    return PBnJMaker.spreadJam();
11 }).then(() => {
12    return PBnJMaker.closeSandwich();
13 }).then((result) => {
14    console.log("Eat that " + result);
15 });
```

Result

```
}).then(() => {
    return PBnJMaker.closeSandwich();
```

@joel__lord
#iJS18

Code

```
1 let PBnJMaker = require("./pbnjmaker");
2
3 PBnJMaker.fetchBread().then(() => {
4     return PBnJMaker.fetchPeanutButter();
5 }).then(() => {
6     return PBnJMaker.fetchJam();
7 }).then(() => {
8     return PBnJMaker.spreadPeanutButter();
9 }).then(() => {
10    return PBnJMaker.spreadJam();
11 }).then(() => {
12    return PBnJMaker.closeSandwich();
13 }).then((result) => {
14    console.log("Eat that " + result);
15 });
```

Result

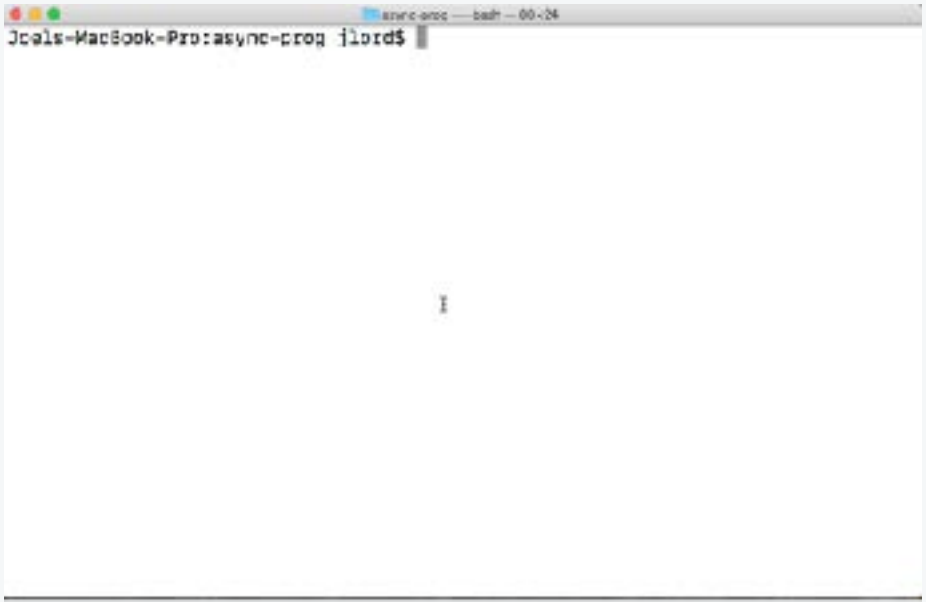
```
}).then((result) => {
    console.log("Eat that " + result);
```

@joel__lord
#iJS18

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
#iJS18

Code

```
1 let PBnJMaker = require('./pbnjmaker');
2
3 PBnJMaker.simulateError();
4
5 PBnJMaker.fetchBread().then(() => {
6     return PBnJMaker.fetchPeanutButter();
7 }).then(() => {
8     return PBnJMaker.fetchJam();
9 }).catch((err) => {
10     console.log('Error fetching the ingredients');
11 }).then(() => {
12     return PBnJMaker.spreadPeanutButter();
13 }).then(() => {
14     return PBnJMaker.spreadJam();
15 }).catch((err) => {
16     console.log('An error occurred while preparing the sandwich');
17 }).then(() => {
18     return PBnJMaker.closeSandwich();
19 }).then((result) => {
20     console.log('Eat that ' + result);
21 }).catch((err) => {
22     console.log('An error occurred');
23 });
```

Result

```
}).catch((err) => {
    console.log("Error with ingredients");
```

@joel__lord
#iJS18

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

```
}).catch((err) => {
    console.log("Error with ingredients");
```

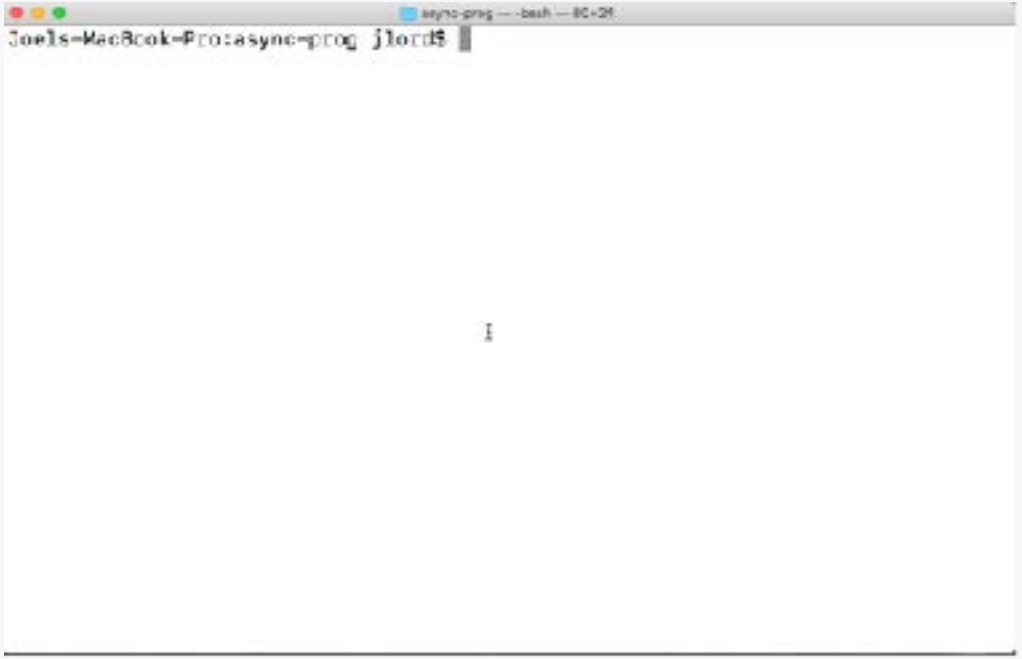
```
}).catch((err) => {
    console.log("Error preparing");
```

```
}).catch((err) => {
    console.log("Error somewhere");
```

@joel__lord
#iJS18

Result

Promises



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

```
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

```
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

```
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

```
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

```
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

```
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?

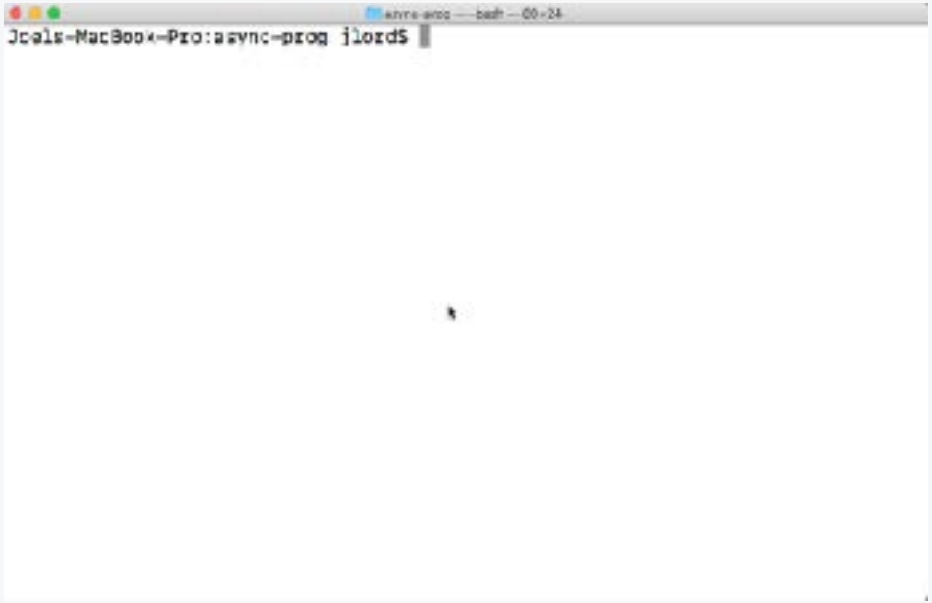
- An iterable function
- “Pauses” the execution of a function

```
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



Async Generators

- Combining the power of promises and generators

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

    function handle(result) {
        // generator returns {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);
});
```

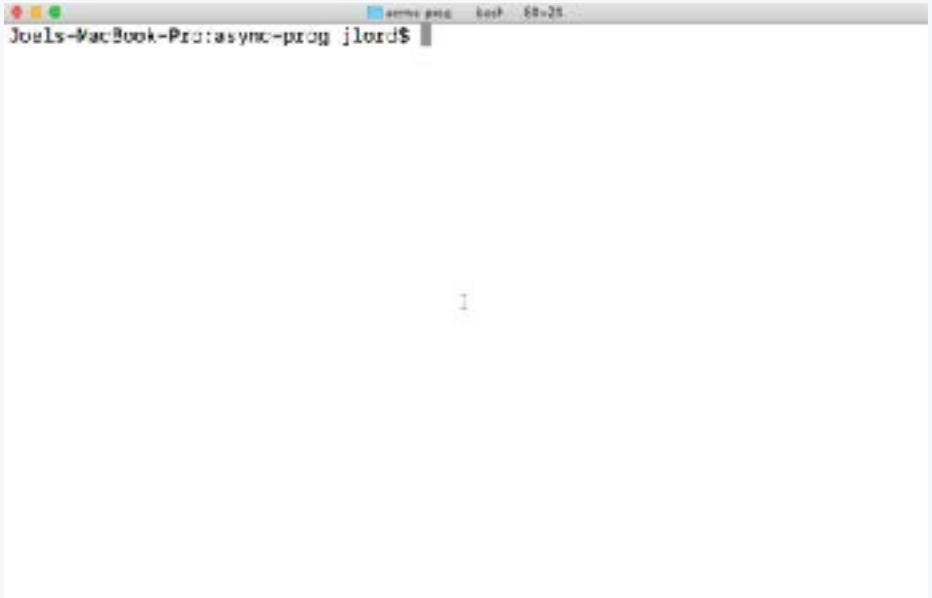
Result

@joel__lord
#iJS18

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
#iJS18


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

```
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?

- 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 = …

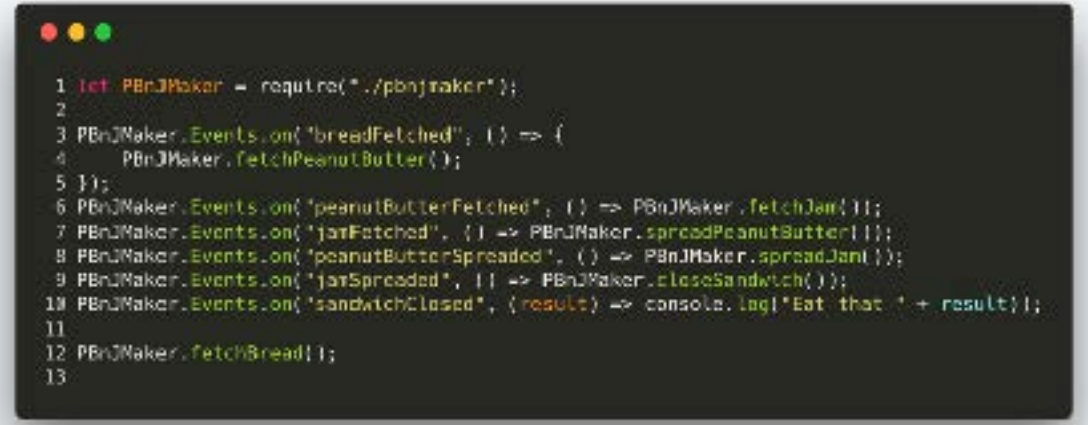
Node Event Emitter

- 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

```
1 let PBnJMaker = require('./pbnjmaker');
2
3 PBrJMaker.Events.on("breadFetched", () => {
4     PBnJMaker.fetchPeanutButter();
5 });
6 PBrJMaker.Events.on("peanutButterFetched", () => PBnJMaker.fetchJam());
7 PBrJMaker.Events.on("jamFetched", () => PBnJMaker.spreadPeanutButter());
8 PBrJMaker.Events.on("peanutButterSpreaded", () => PBnJMaker.spreadJam());
9 PBrJMaker.Events.on("jamSpreaded", () => PBnJMaker.closeSandwich());
10 PBrJMaker.Events.on("sandwichClosed", (result) => console.log('Eat that ' + result));
11
12 PBrJMaker.fetchBread();
13
```

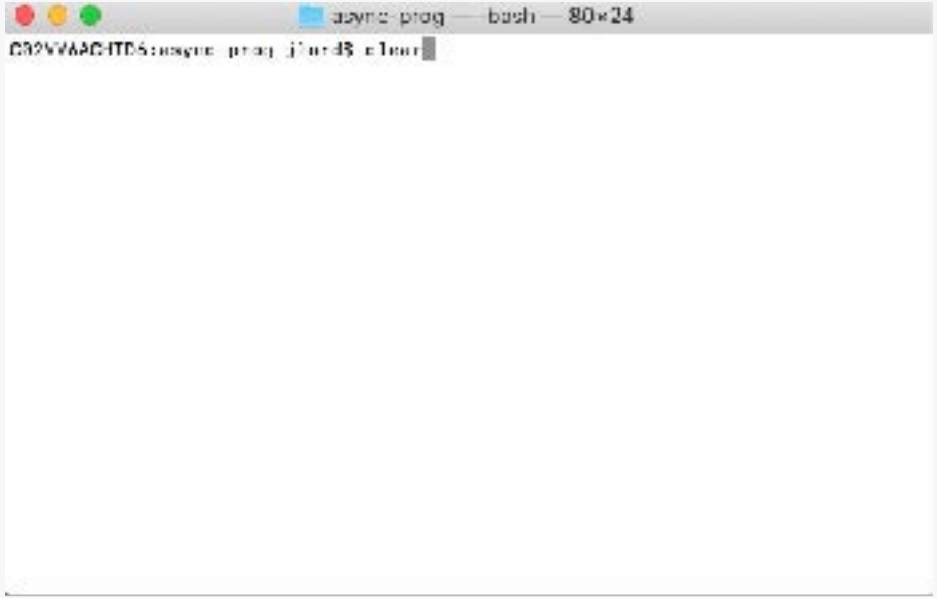
Demo

@joel__lord
#iJS18

Code

```
const 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.clapSandwich();
});

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

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

Demo



@joel__lord
#iJS18


Reactive Stream
(2017)

RxJS

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

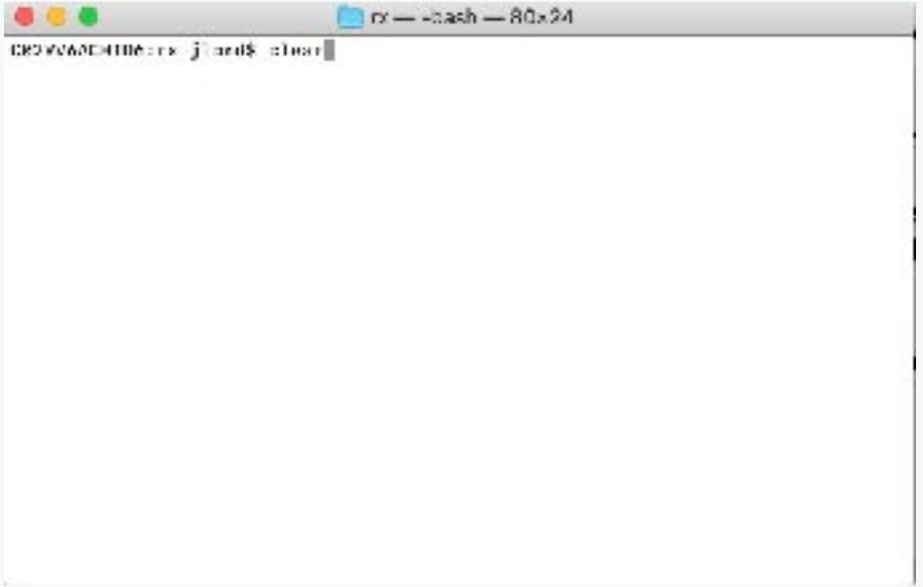
```
getDataFromLocalMemory()
    .filter (s => s != null)
    .map(s => `${s} transformed`)
    .forEach(s => console.log(`next => ${s}`))
```

```
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 });
```

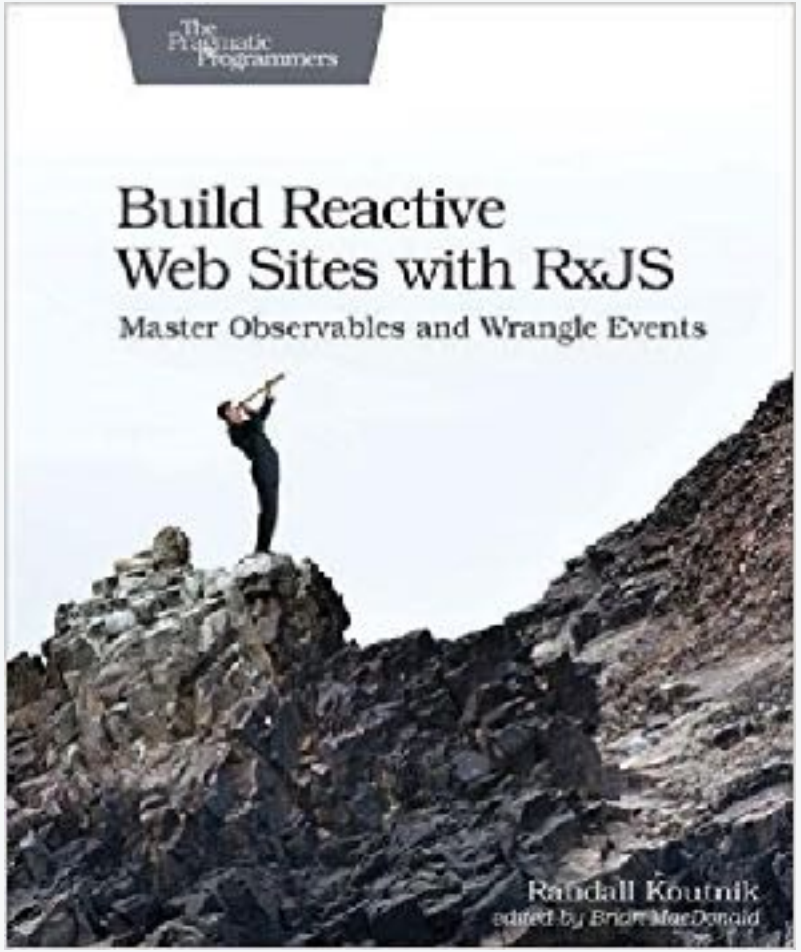
Demo



@joel__lord
#iJS18

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

iJS, London, UK
April 11, 2018

Special thanks to
@tomasz_ducin for the inspiration


@joel__lord


joellord