



Asynchronicity: concurrency. A tale of

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

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

```
<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
```

```
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

```
<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
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

```
<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  
  
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

```
<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  
  
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

```
<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>
```

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

```
<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  
  
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

```
<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  
  
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

```
<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  
  
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

```
<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
```

```
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

```
<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
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

```
<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
```

```
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

```
<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  
  
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

```
<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  
  
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

```
<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  
  
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

```
<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  
  
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

```
<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

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

```
<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  
  
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

```
<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  
  
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

```
<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

A photograph featuring a turtle and five rabbits. The turtle is positioned at the top center, facing upwards. Below it, five rabbits are arranged in a row, also facing upwards. The rabbits are white with black spots and ears. The background is a solid red color with vertical white stripes. A dark grey horizontal bar is overlaid across the middle of the image, containing the text "Race Conditions" in white.

Race Conditions

About Me



@joel__lord



joellord

WARNING

WARNING

175 slides



@joel__lord
#iJS18

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!





The Event Loop

Classic Architecture (PHP, Java)



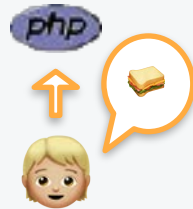
Classic Architecture (PHP, Java)

- 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
 -



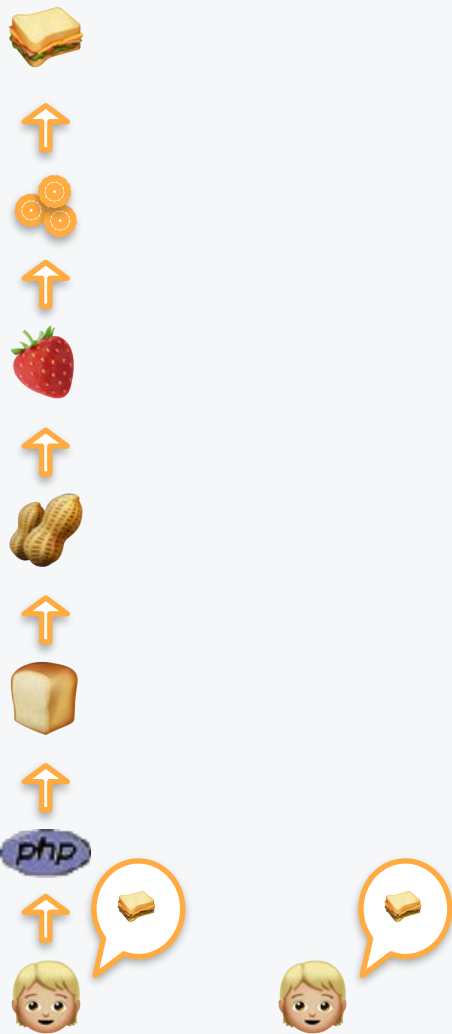
Classic Architecture (PHP, Java)

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



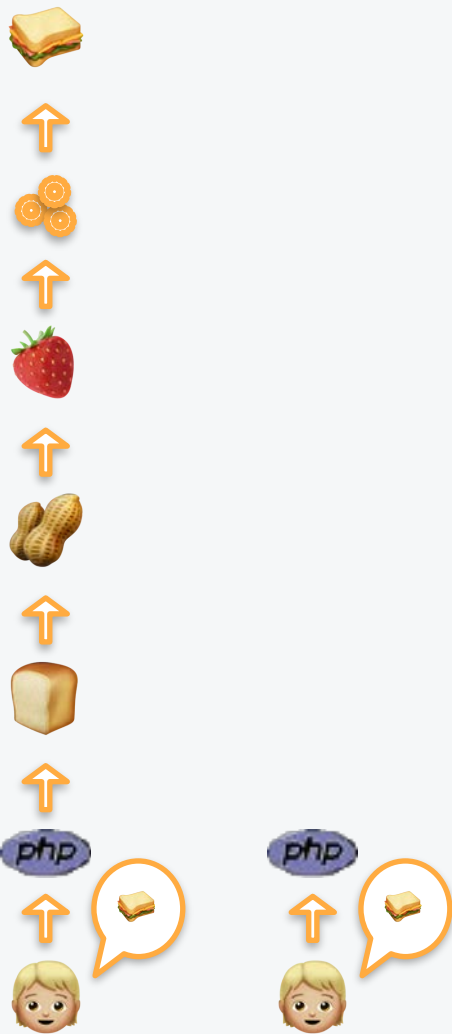
Classic Architecture (PHP, Java)

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



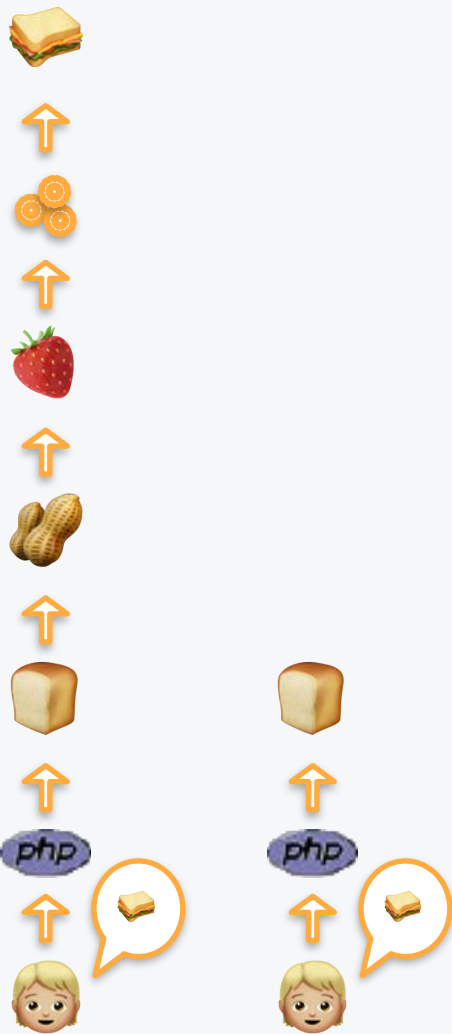
Classic Architecture (PHP, Java)

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



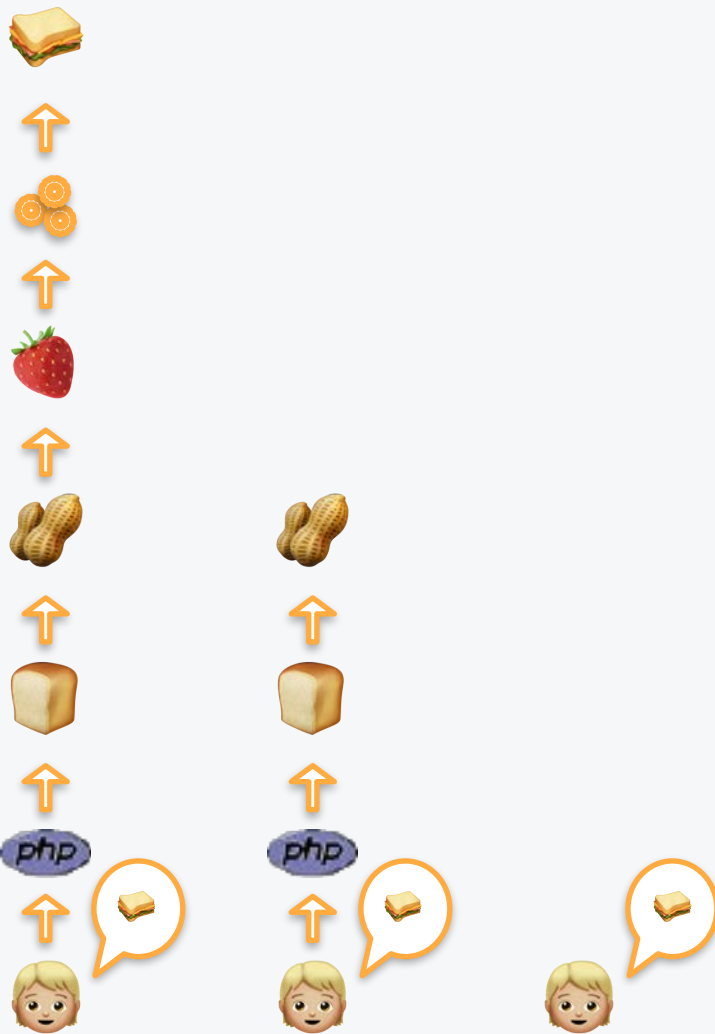
Classic Architecture (PHP, Java)

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



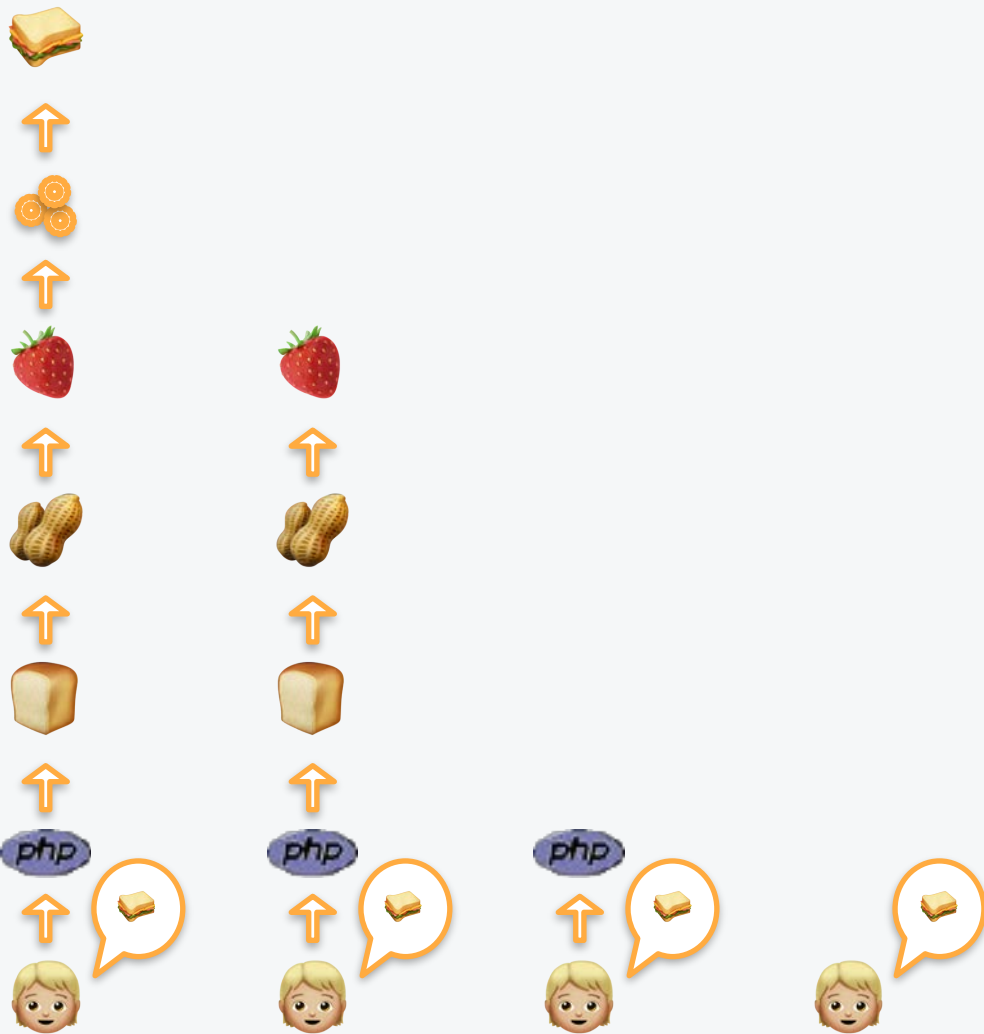
Classic Architecture (PHP, Java)

- 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
-



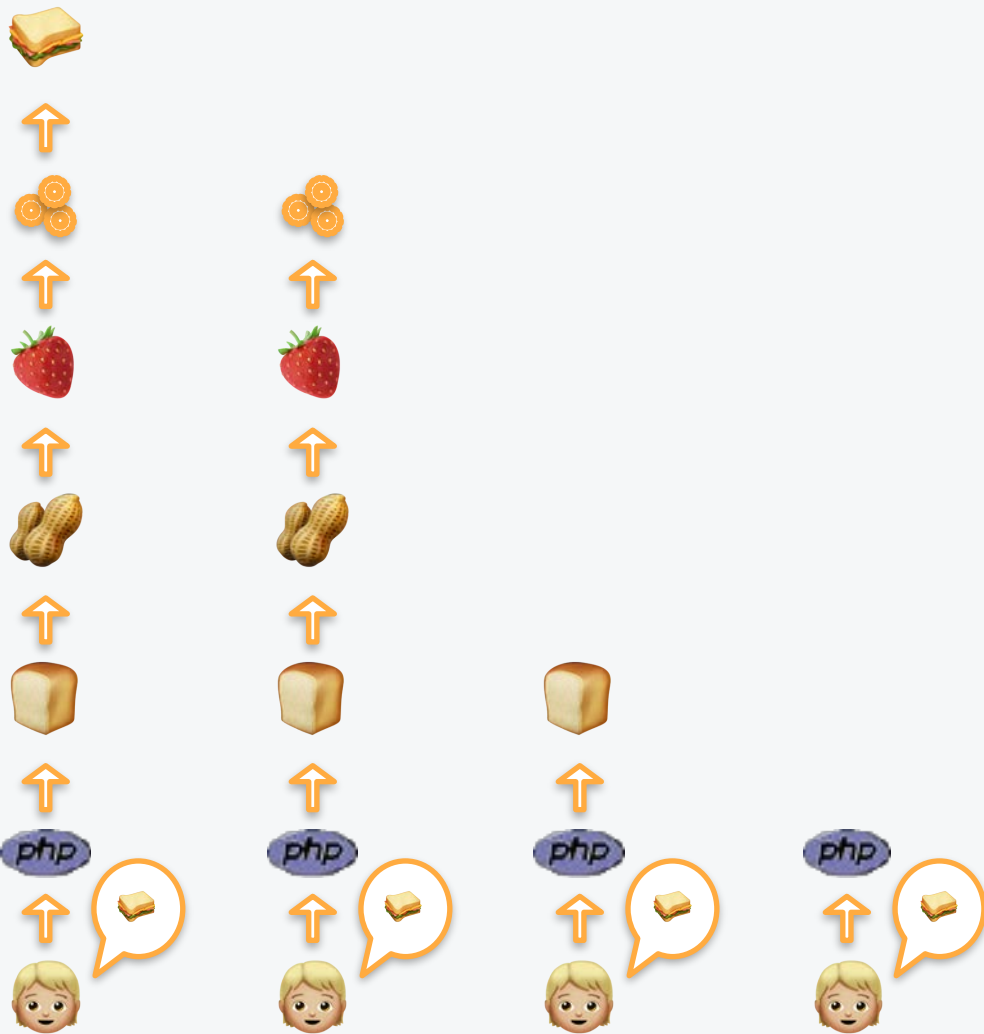
Classic Architecture (PHP, Java)

- 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



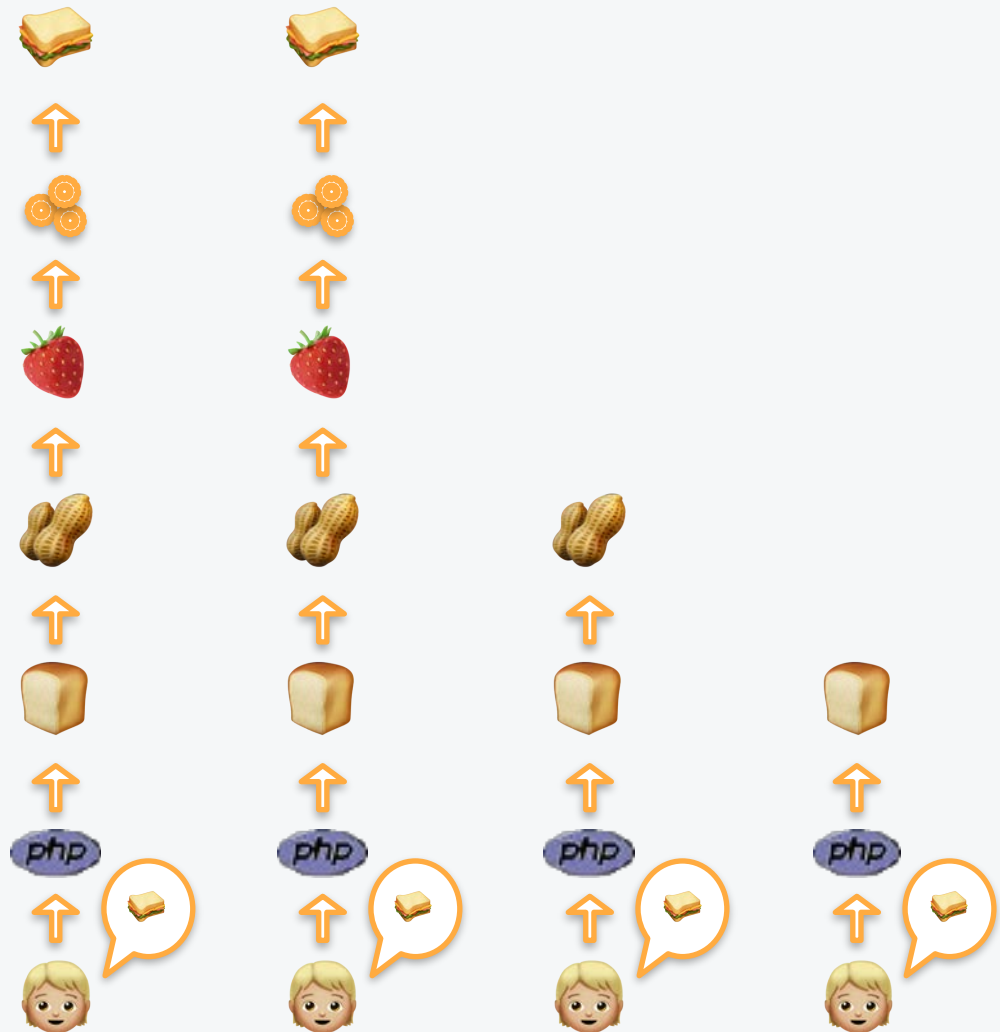
Classic Architecture (PHP, Java)

- 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



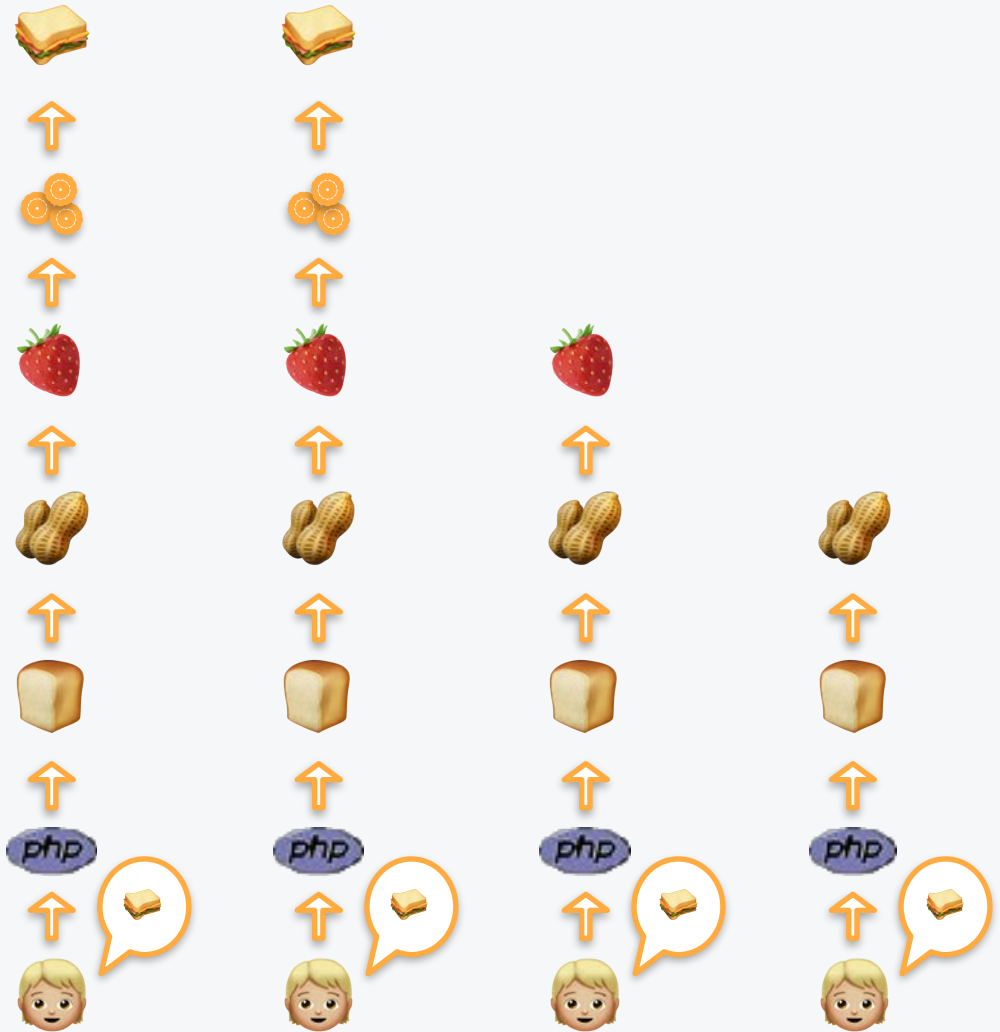
Classic Architecture (PHP, Java)

- 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



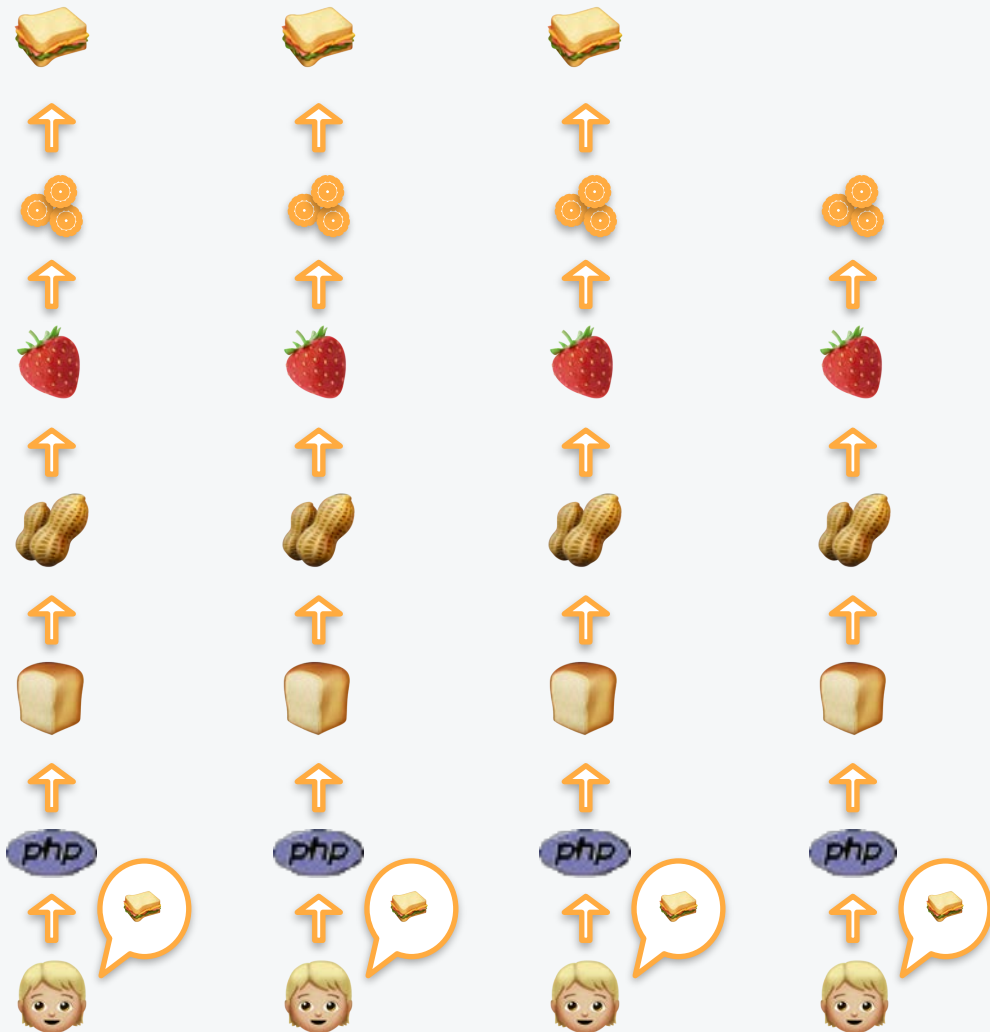
Classic Architecture (PHP, Java)

- 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



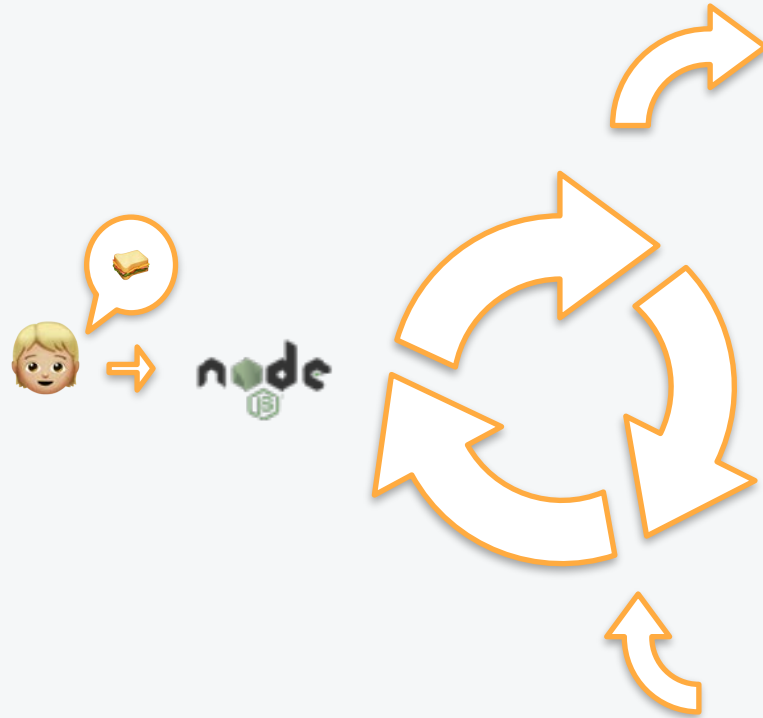
Classic Architecture (PHP, Java)

- 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



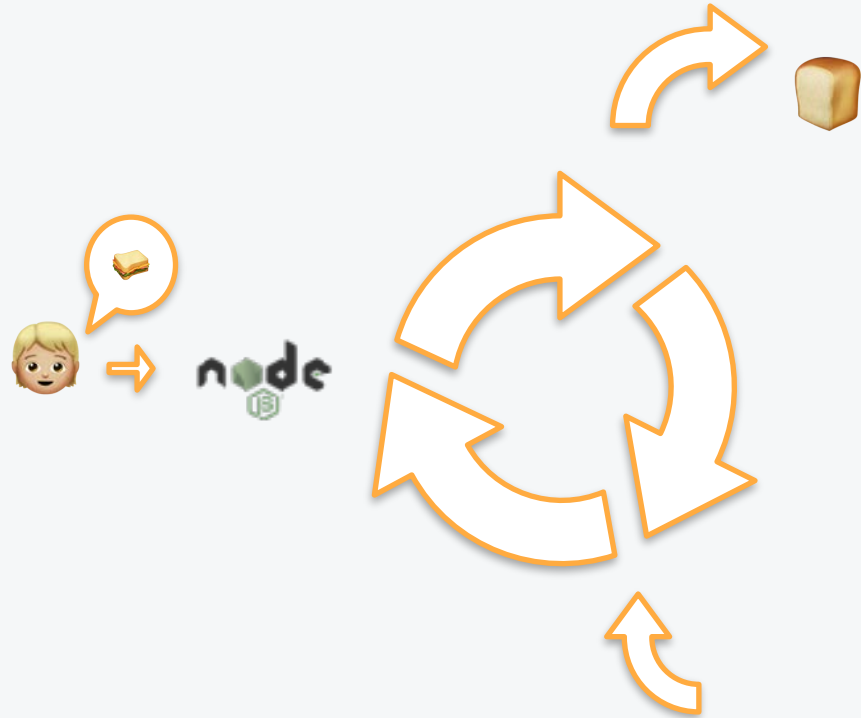
Event Loop (NodeJS)

- 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



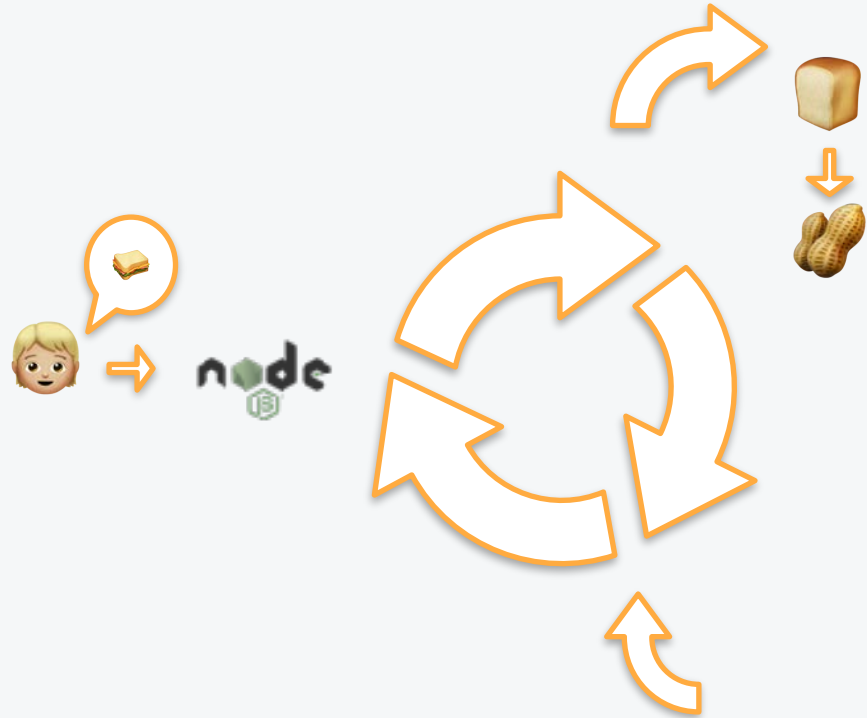
Event Loop (NodeJS)

- 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



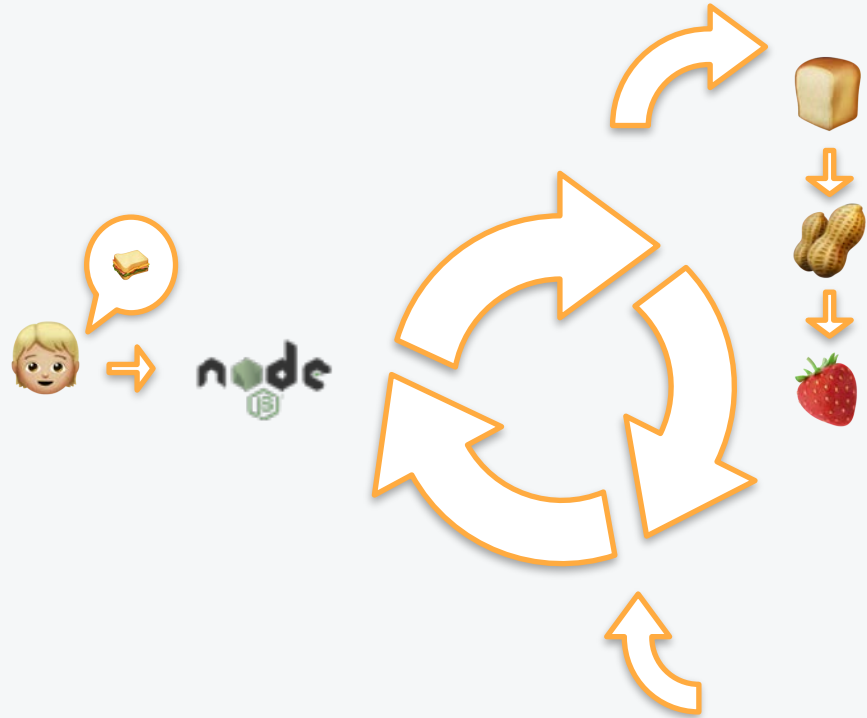
Event Loop (NodeJS)

- 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



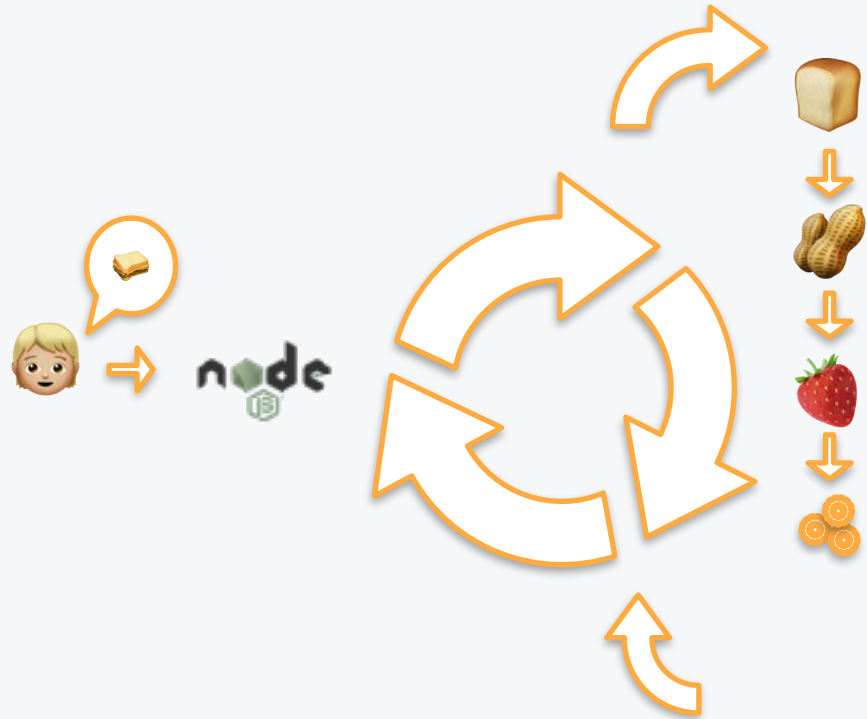
Event Loop (NodeJS)

- 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



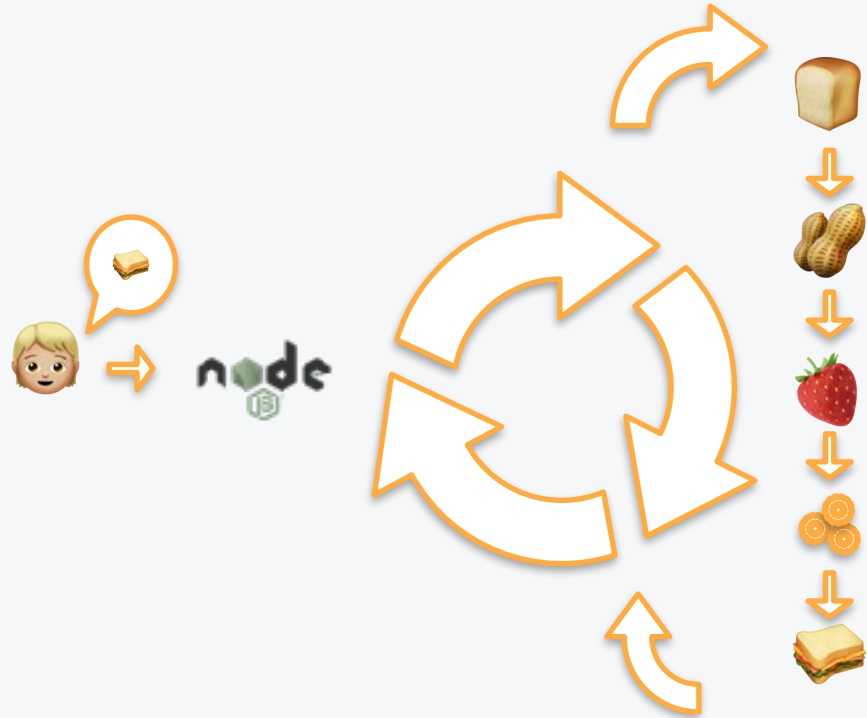
Event Loop (NodeJS)

- 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



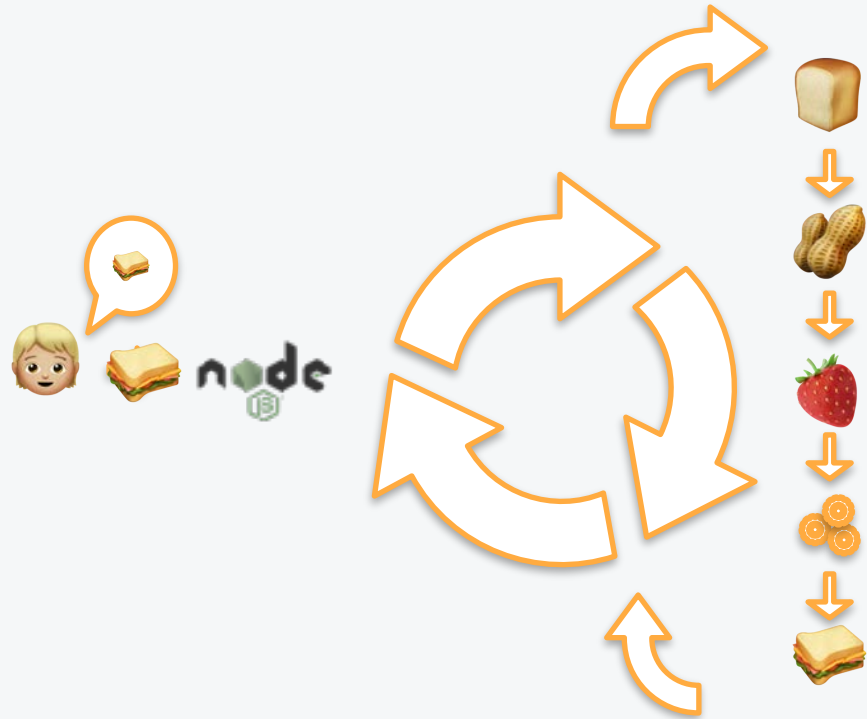
Event Loop (NodeJS)

- 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



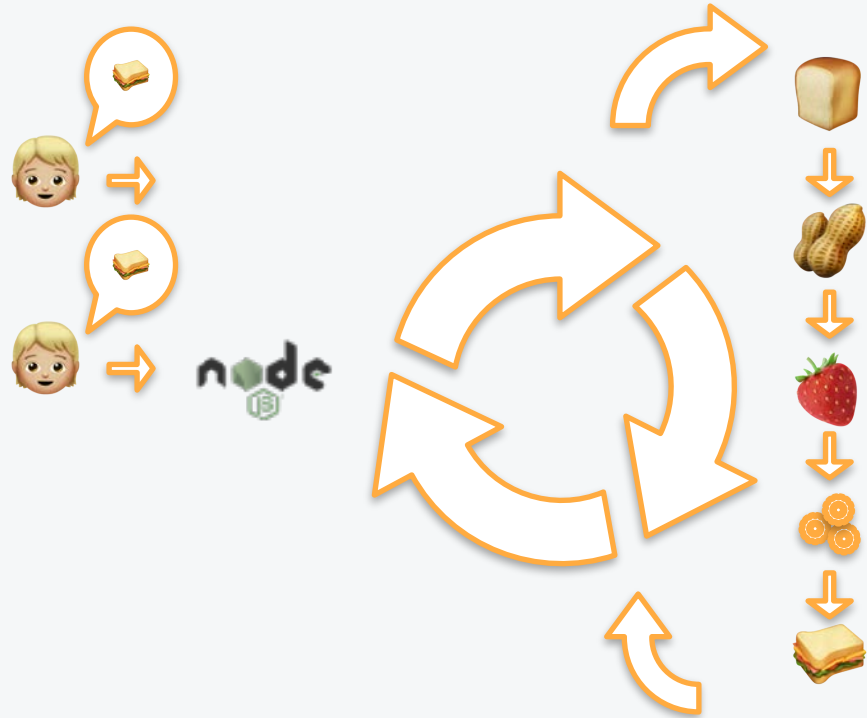
Event Loop (NodeJS)

- 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



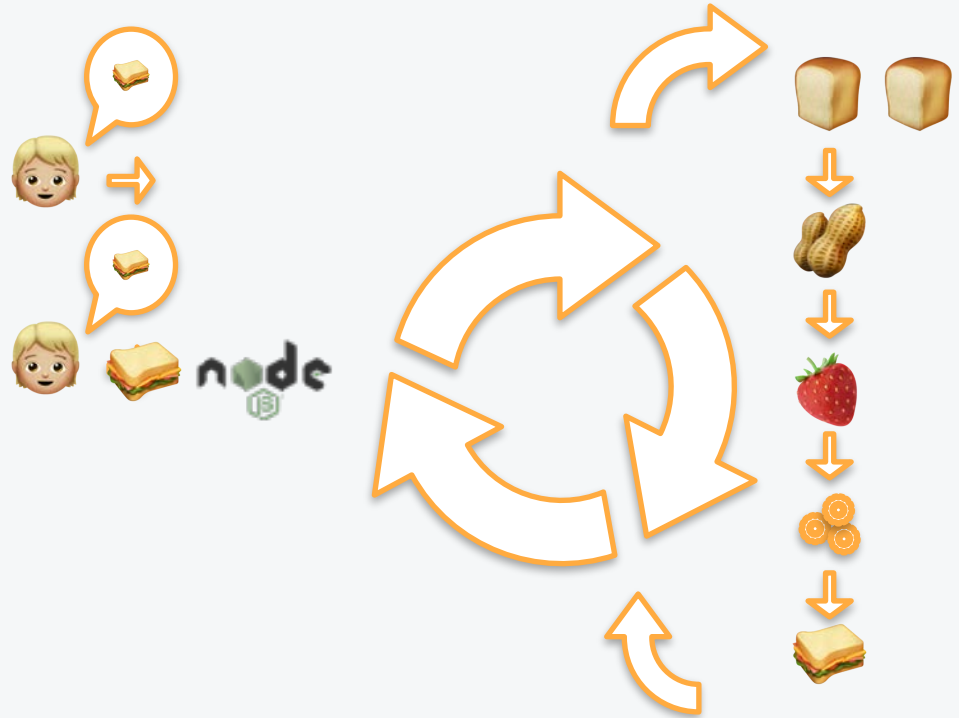
Event Loop (NodeJS)

- 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



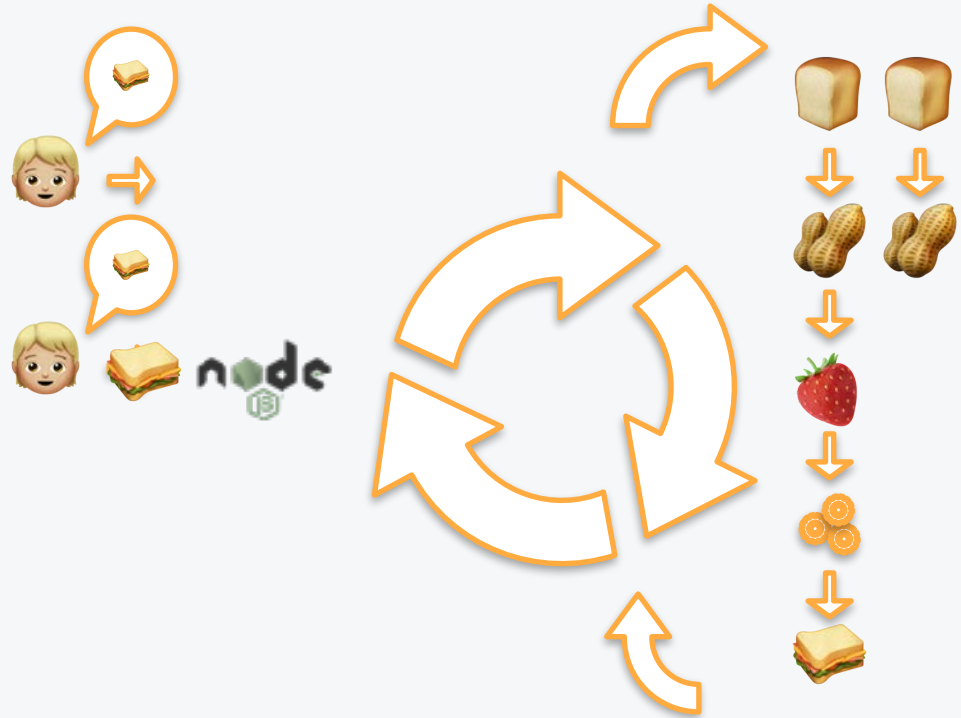
Event Loop (NodeJS)

- 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



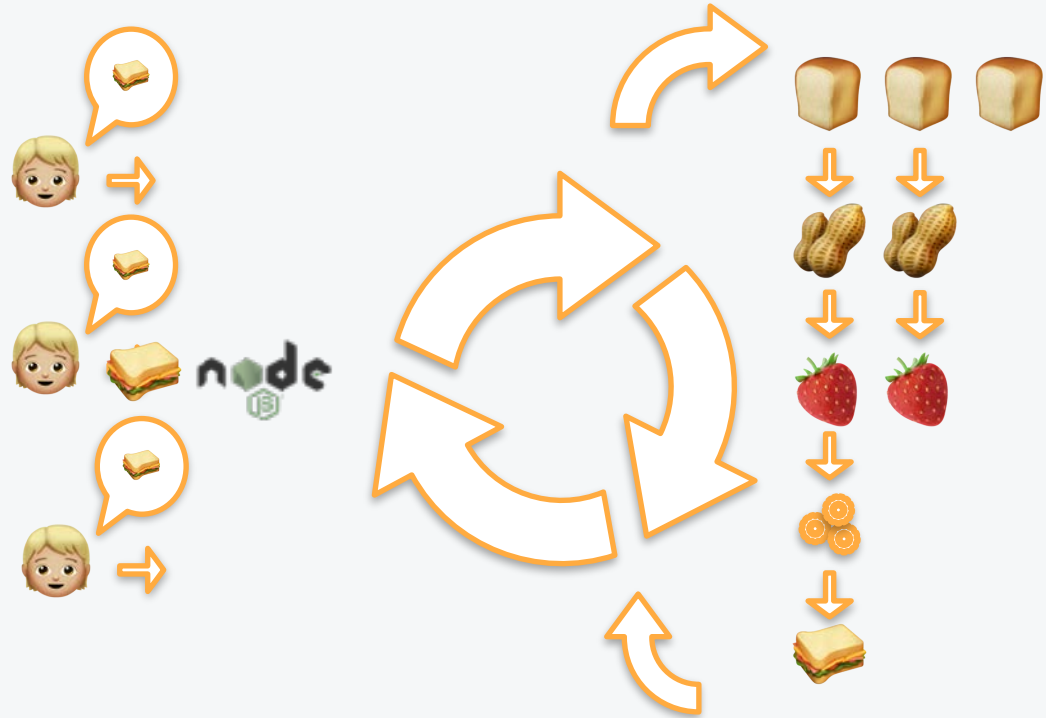
Event Loop (NodeJS)

- 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



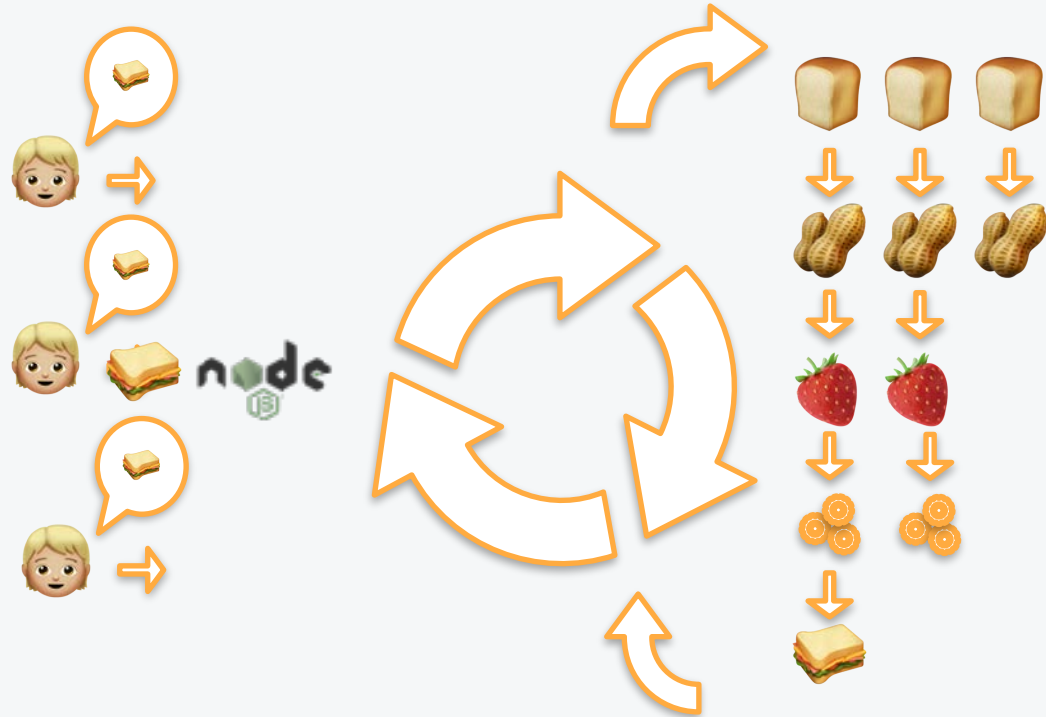
Event Loop (NodeJS)

- 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



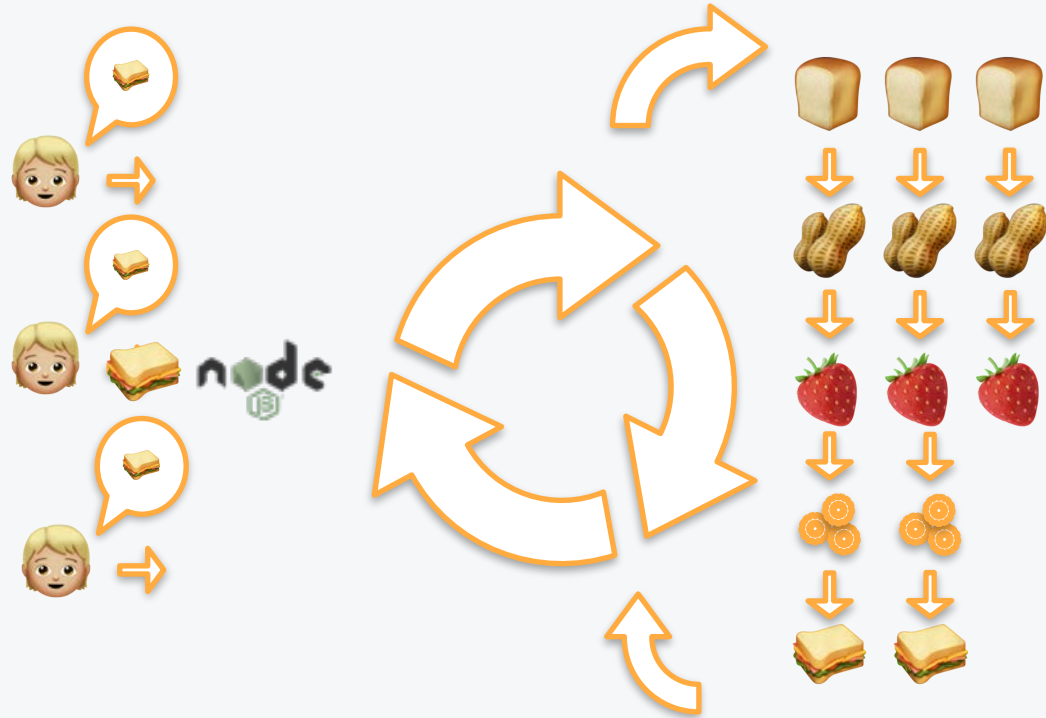
Event Loop (NodeJS)

- 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



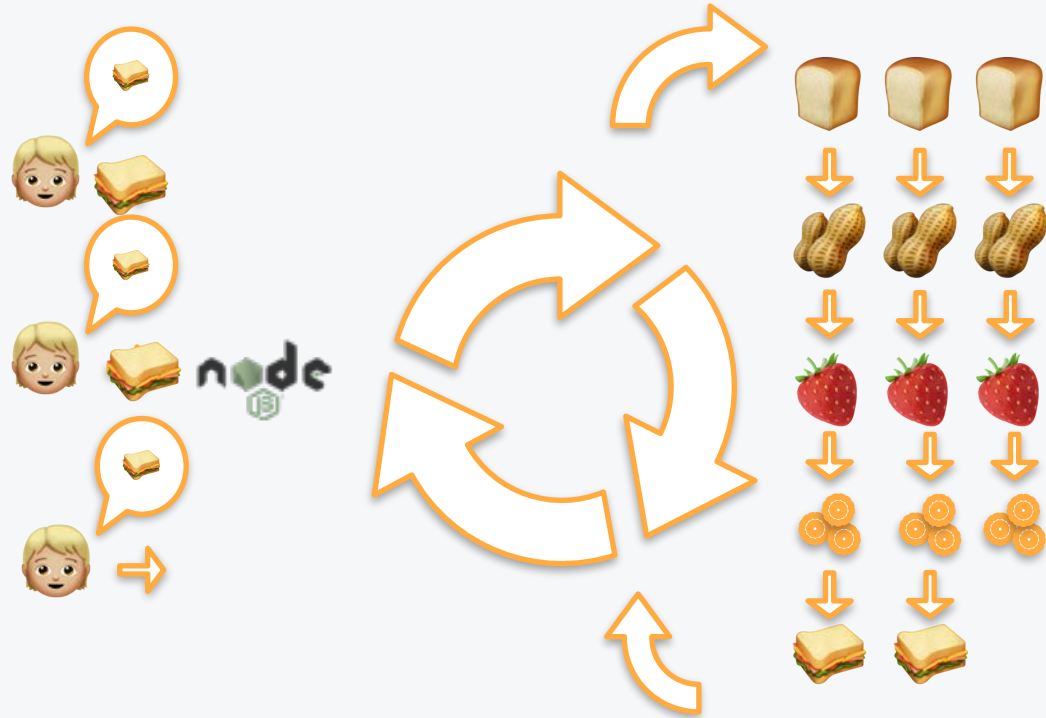
Event Loop (NodeJS)

- 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



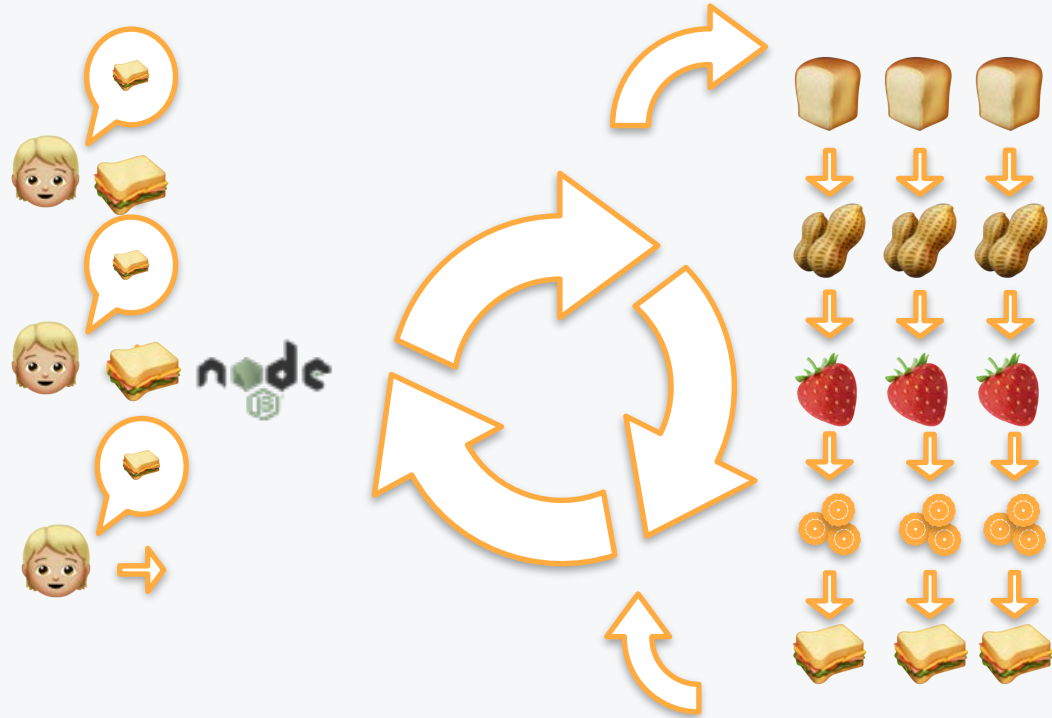
Event Loop (NodeJS)

- 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



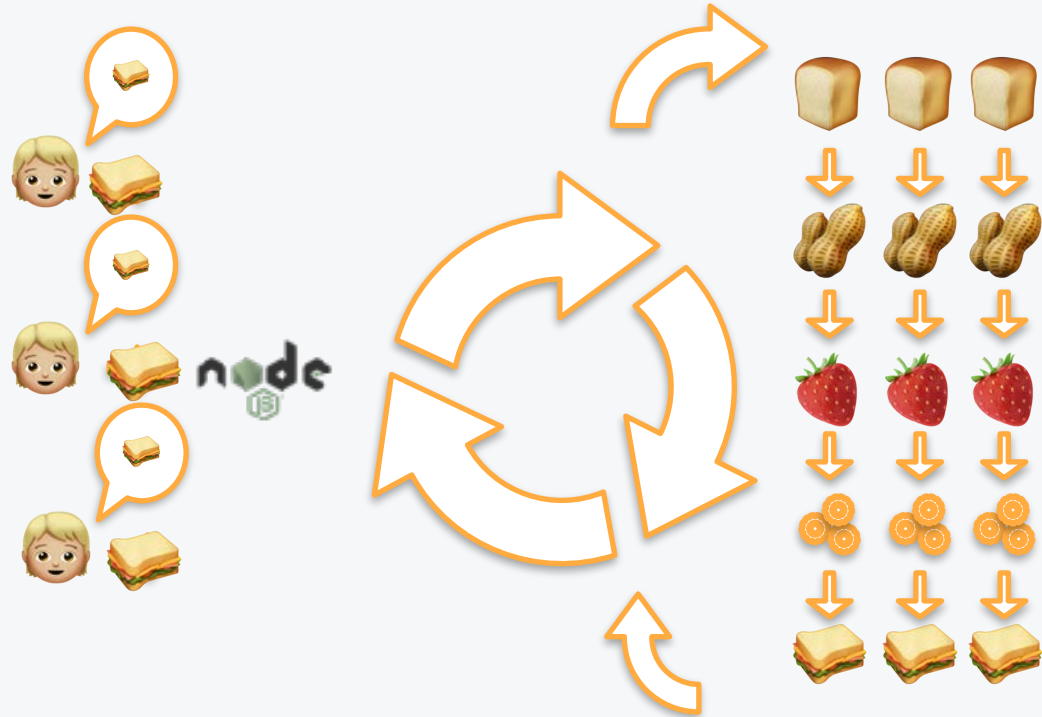
Event Loop (NodeJS)

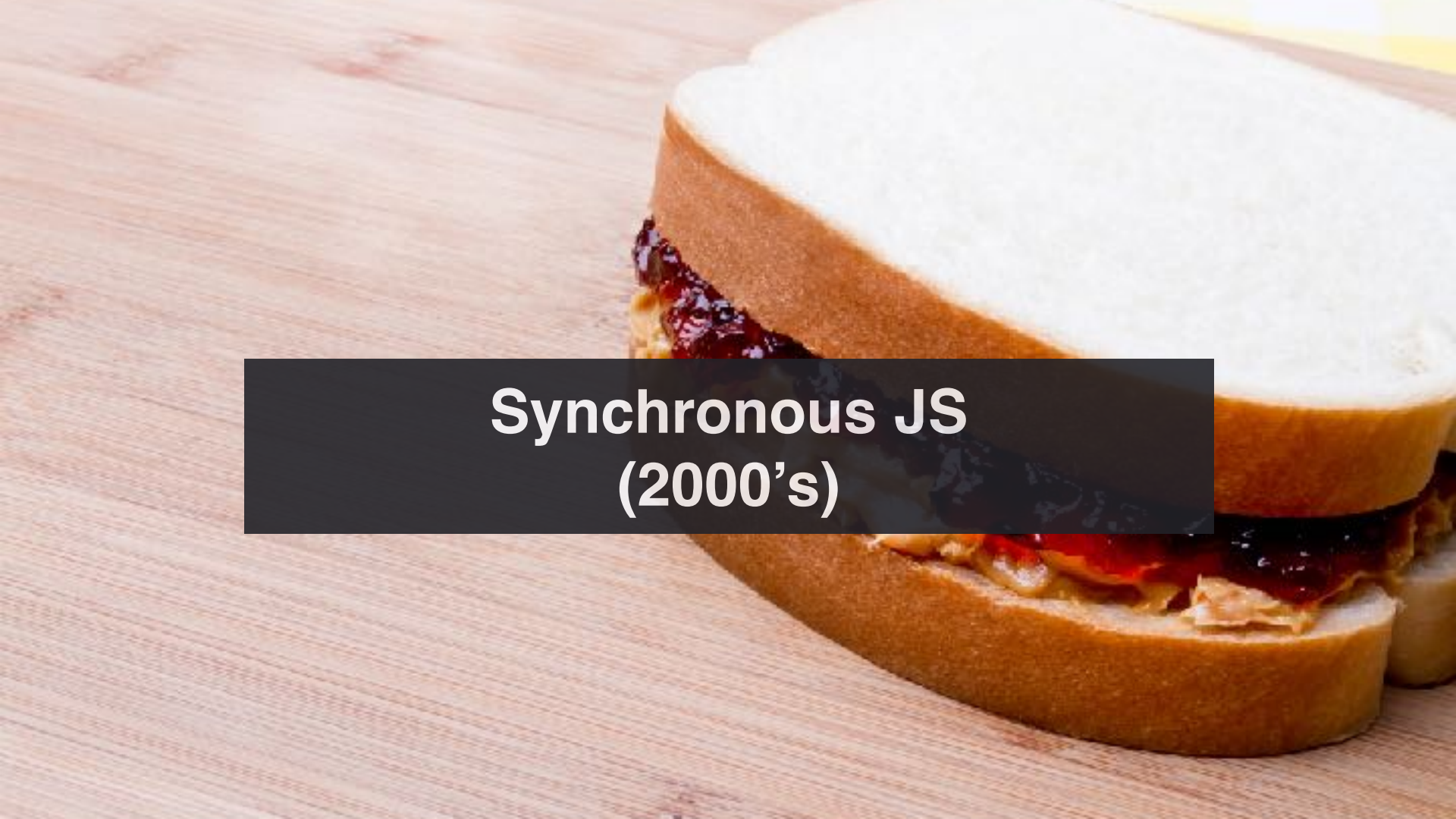
- I want a sandwich!
- Sure
 - “Worker, Make a sandwich”
- A new customer arrives:
I want a sandwich!
- Sure
 - “Other worker,
make a sandwich”
- Here is your sandwich
customer 1



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





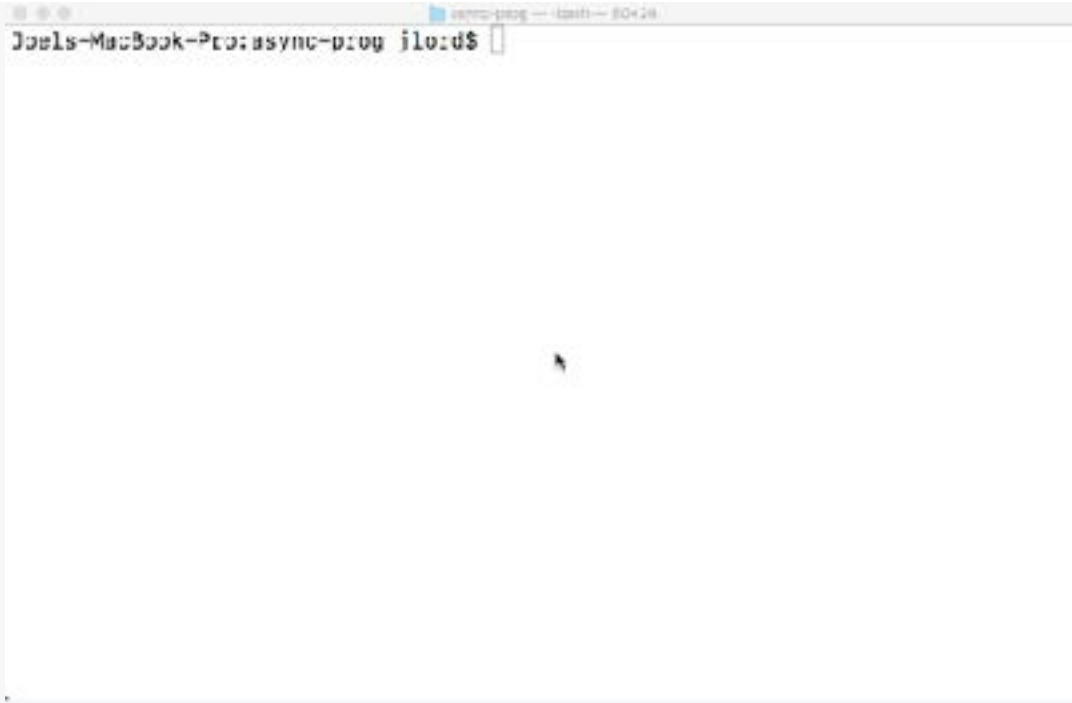
**Synchronous JS
(2000's)**

Code

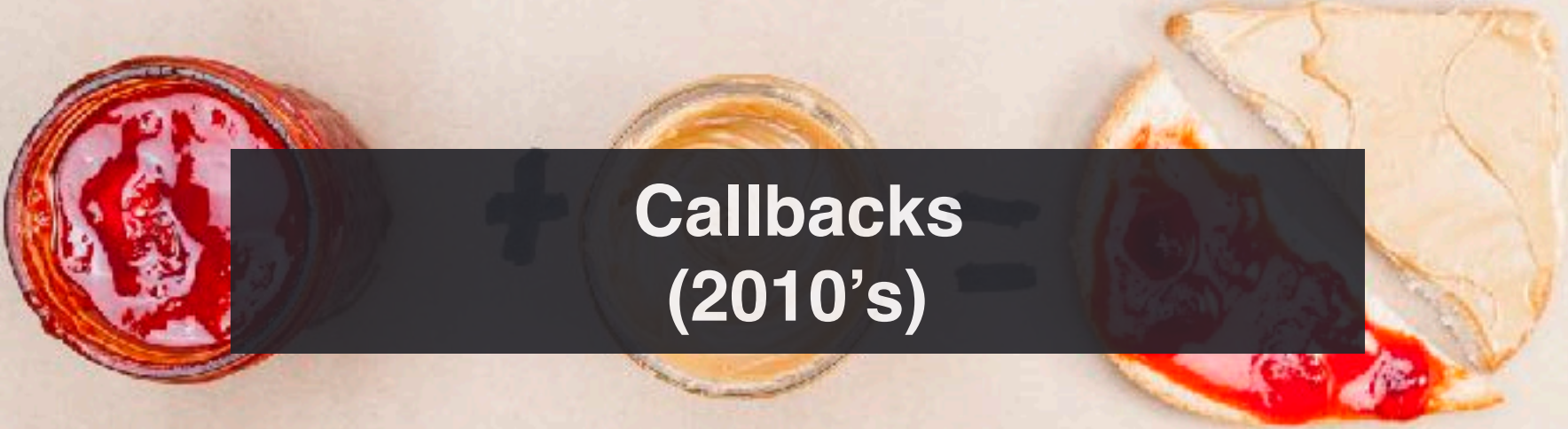
```
let PBnJMaker = require("./pbnjmaker");

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

Result



A screenshot of a terminal window on a Mac. The window title is "async-prog -- bash -- 80x24". The prompt is "Joels-MacBook-Pro: async-prog joelord\$". The terminal is otherwise empty.



**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

A terminal window titled 'async-prog -- bash -- 80x25' is shown. The prompt is 'Joels-MacBook-Pro:async-prog jlord\$'. The user enters 'node ./examples/callbacks.js'. The output is 'begin', followed by '1', '2', and '3' on separate lines. The prompt returns to 'Joels-MacBook-Pro:async-prog jlord\$' with a cursor.

```
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
1
2
3
after
Joels-MacBook-Pro:async-prog jlord$
```

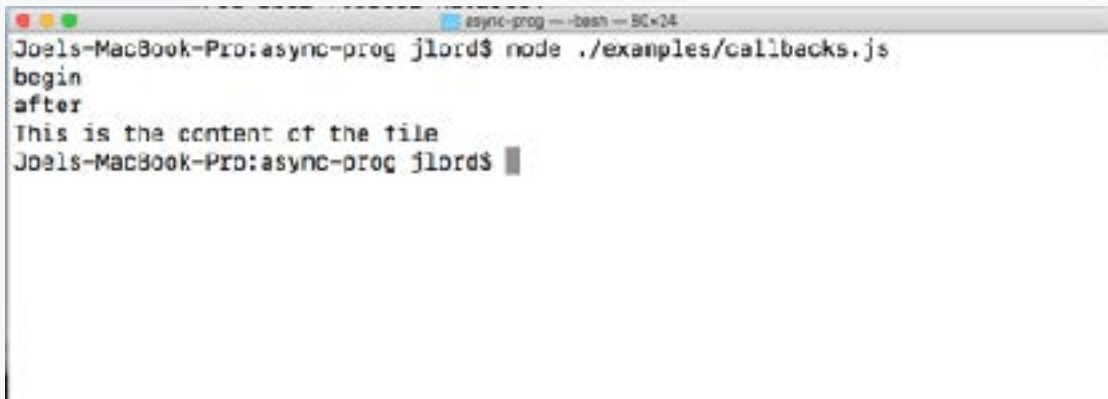
What is a callback

- 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



```
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$
```


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

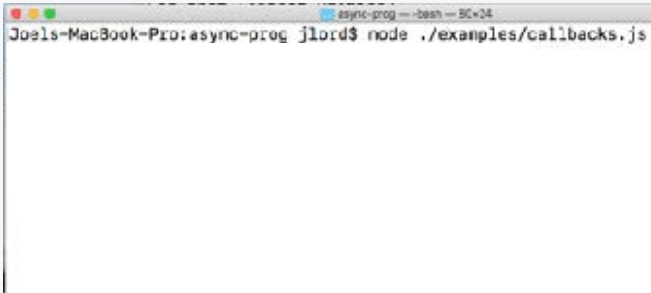
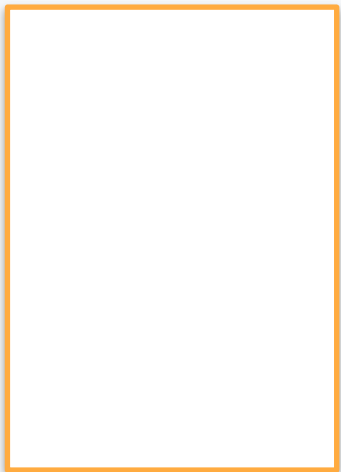
- Simply a function as an argument to a function
- Could be synchronous or asynchronous

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

What is a callback

Asynchronous callback and the event loop

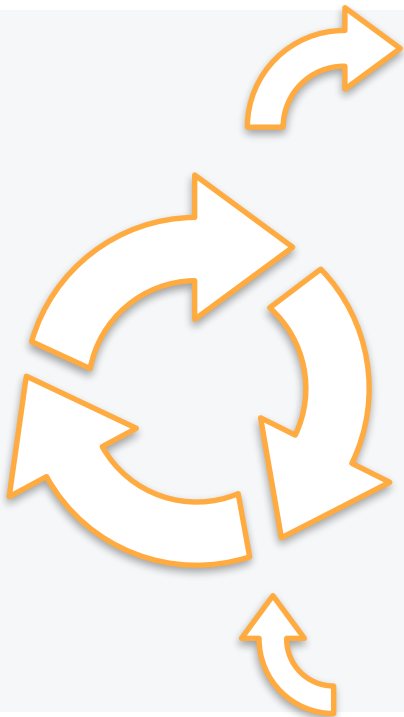
```
console.log("begin");  
fs.readFile("./cb.txt",  
(err, data) => {  
  console.log(data);  
});  
console.log("after");
```



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
```

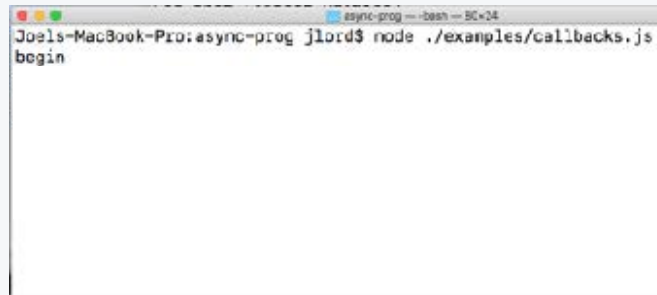
What is a callback

Asynchronous callback and the event loop

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

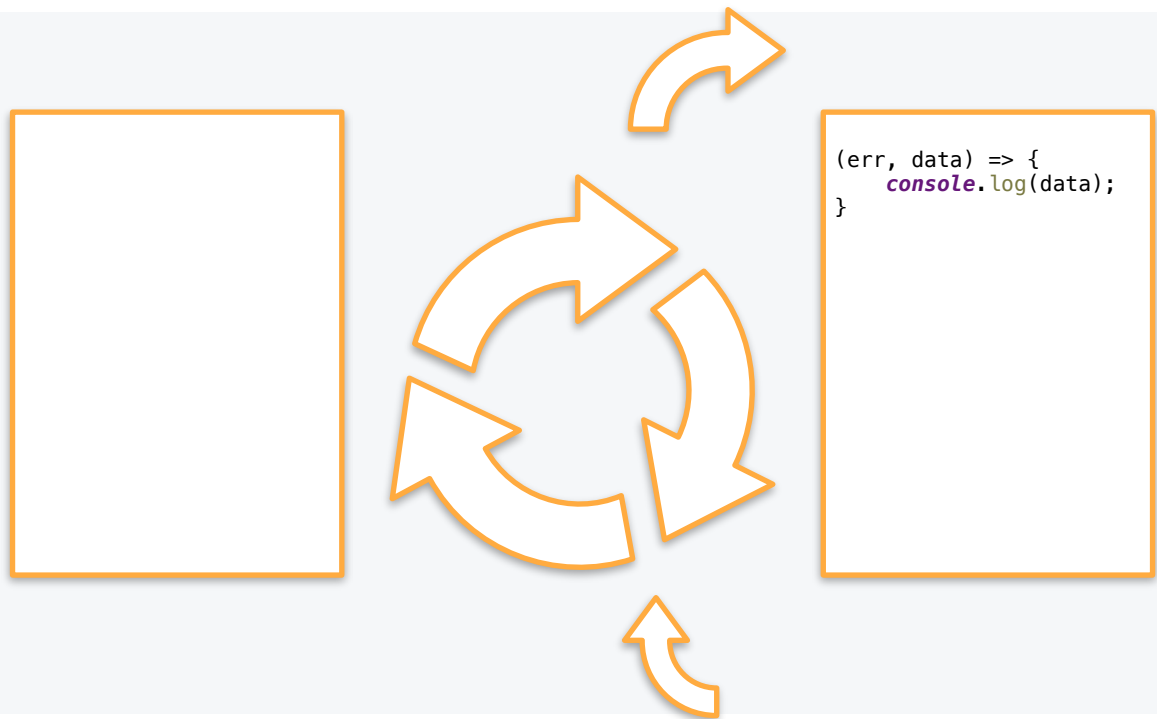


```
(err, data) => {  
  console.log(data);  
}
```



What is a callback

Asynchronous callback and the event loop

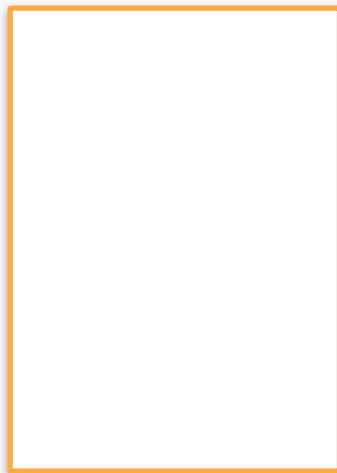


```
async-prog --bash --BC=14
Joels-MacBook-Pro:async-prog jlord$ node ./examples/callbacks.js
begin
```

What is a callback

Asynchronous callback and the event loop

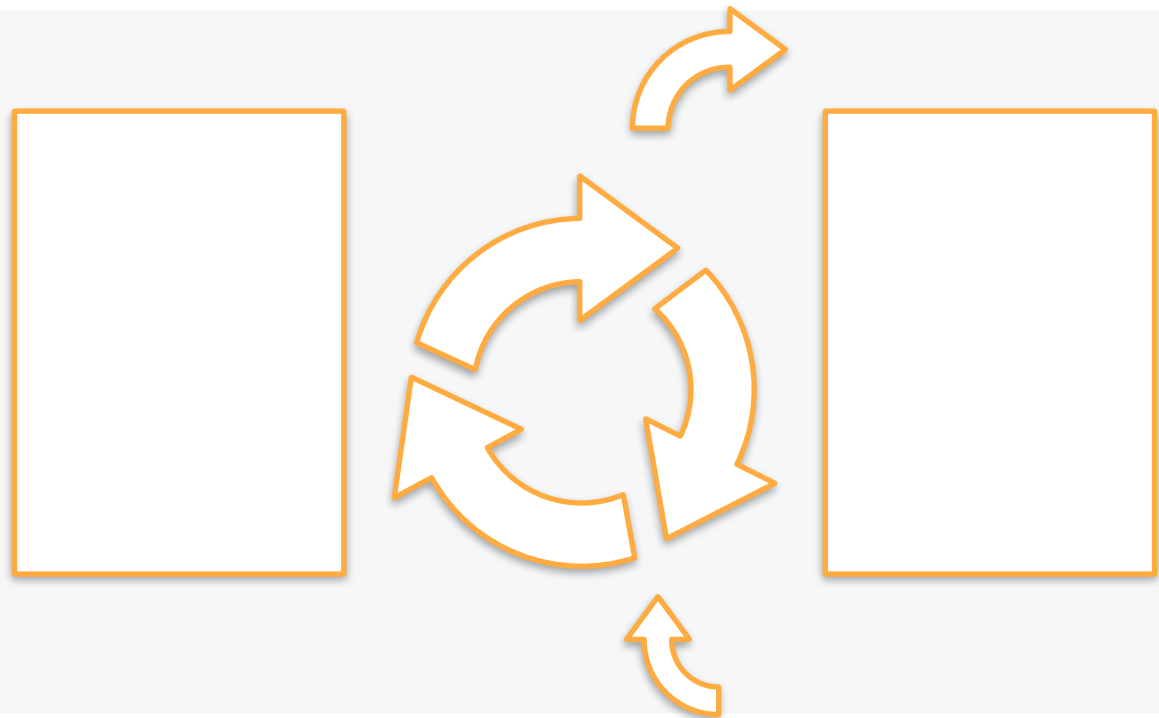
```
(err, data) => {  
  console.log(data);  
}
```



```
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$
```



@joel__lord
#iJS18

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

A screenshot of a terminal window on a Mac. The window title is "async-prog -- test -- 88x25". The prompt is "Joels-MacBook-Pro:async-prog jlord\$". The terminal is otherwise empty.

```
Joels-MacBook-Pro:async-prog jlord$
```

Callbacks

- It works!
- but...

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

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");

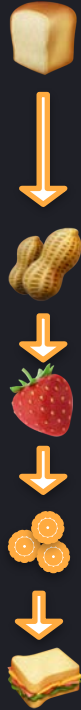
PBnJMaker.simulateError();

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) => {
```

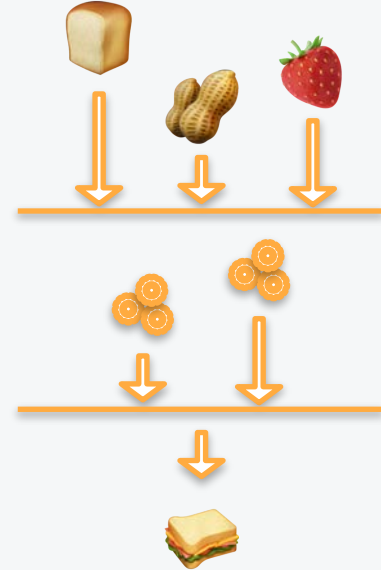
A photograph of a man in a bright blue polo shirt and a young girl in a white floral dress in a kitchen. The man is leaning over the counter, looking at the girl as she uses a knife to spread red jam on a slice of toast. On the counter, there is a plate with another slice of toast, a jar of yellow jam, a jar of dark jam, and a small white lid. The kitchen has wooden cabinets, a sink, and a window with blinds in the background.

Parallel events

Step by step



Parallel events



Code

```
1 let PBnJMaker = require("../pbnjmaker");
2
3 let steps = 0;
4 // PBnJMaker, simulateError();
5
6 function fetchIngredients() {
7   PbnMaker.fetchIngredients();
8   PbnMaker.fetchPeanutButter(fetchIngredients);
9   PbnMaker.fetchJam(fetchIngredients);
10 }
11
12 function handleIngredients(err) {
13   if (err) return console.log("An error occurred while fetching the ingredients");
14   ingredients++;
15   if (ingredients === 3) {
16     prepareSandwich();
17   }
18 }
19
20 function prepareSandwich() {
21   PbnMaker.spreadPeanutButter(preparationStepCompleted);
22   PbnMaker.spreadJam(preparationStepCompleted);
23 }
24
25 function preparationStepCompleted(err) {
26   if (err) console.log("An error occurred while preparing the sandwich");
27   steps++;
28   if (STEPS === 3) {
29     completeSandwich();
30   }
31 }
32
33 function completeSandwich() {
34   PbnMaker.closeSandwich(err, result) => {
35     if (err) {
36       console.log("An error occurred");
37     } else {
38       console.log("Eat that " + result);
39     }
40   };
41 }
42
43 fetchIngredients();
```

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



@joel__lord
#JS18

Code

```
1 // Promise.all() example - Promise.all()
2 let ingredients = 5;
3 let steps = 0;
4 // P1: Baker, spreadButter()
5
6
7 function fetchIngredients() {
8   // P1: Baker, spreadButter() handles ingredients;
9   P1: Baker, fetchPeanutButter(fetchIngredients);
10  P1: Baker, fetchJam(fetchIngredients);
11 }
12
13 function handleIngredients(err) {
14   if (err) return console.log("An error occurred while fetching the ingredients");
15   ingredients++;
16   if (ingredients === 3) {
17     prepareSandwich();
18   }
19 }
20
21 function prepareSandwich() {
22   P1: Baker, spreadPeanutButter(fetchIngredients);
23   P1: Baker, spreadJam(preparationStepCompleted);
24 }
25
26 function preparationStepCompleted(err) {
27   if (err) console.log("An error occurred while preparing the sandwich");
28   steps++;
29   if (steps === 2) {
30     completeSandwich();
31   }
32 }
33
34 function completeSandwich() {
35   P1: Baker, closeSandwich(err, result) => {
36     if (err) {
37       console.log("An error occurred");
38     } else {
39       console.log("Eat that " + result);
40     }
41   };
42 }
43 fetchIngredients();
```

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



@joel__lord
#iJS18

Code

```
1 let PJsMaker = require('./pjsmaker');
2 let ingredients = 8;
3 let steps = 0;
4 // PJsMaker, completeSteps();
5 function fetchIngredients() {
6   PJsMaker.fetchPeanutButter(fetchIngredients);
7   PJsMaker.fetchJam(fetchIngredients);
8 }
9
10
11
12 function handleIngredients(err) {
13   if (err) return console.log("An error occurred while fetching the ingredients");
14   ingredients++;
15   if (ingredients === 3) {
16     prepareSandwich();
17   }
18 }
19
20 function prepareSandwich() {
21   PJsMaker.spreadPeanutButter(preparationStepCompleted);
22   PJsMaker.spreadJam(preparationStepCompleted);
23 }
24
25 function preparationStepCompleted(err) {
26   if (err) console.log("An error occurred while preparing the sandwich");
27   steps++;
28   if (steps === 2) {
29     completeSandwich();
30   }
31 }
32
33 function completeSandwich() {
34   PJsMaker.closeSandwich(err, result) => {
35     if (err) {
36       console.log("An error occurred");
37     } else {
38       console.log("Eat that " + result);
39     }
40   });
41 }
42
43 fetchIngredients();
```

```
function fetchIngredients() {
}
```



@joel__lord
#JS18

Code

```
1 let PBnJMaker = require('./pbnJmaker');
2 let ingredients = 8;
3 let steps = 0;
4 // PBnJMaker.coffeeError();
5
6 // Fetching coffee ☕
7 PBnJMaker.fetchCoffee(handlesIngredients);
8 PBnJMaker.fetchPeanutButter(handlesIngredients);
9 PBnJMaker.fetchJam(handlesIngredients);
10
11
12 function handlesIngredients(err) {
13   if (err) return console.log("An error occurred while fetching the ingredients");
14   ingredients++;
15   if (ingredients === 3) {
16     prepareSandwich();
17   }
18 }
19
20 function prepareSandwich() {
21   PBnJMaker.spreadPeanutButter([preparationStepCompleted]);
22   PBnJMaker.spreadJam([preparationStepCompleted]);
23 }
24
25 function preparationStepCompleted(err) {
26   if (err) steps = console.log("An error occurred while preparing the sandwich");
27   steps++;
28   if (steps === 2) {
29     completeSandwich();
30   }
31 }
32
33 function completeSandwich() {
34   PBnJMaker.closeSandwich(err, result) => {
35     if (err) {
36       console.log("An error occurred");
37     } else {
38       console.log("Eat that " + result);
39     }
40   };
41 }
42
43 fetchIngredients();
```

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



@joel__lord
#iJS18

Code

```
1 let PJsMaker = require('./pjsmaker');
2 let ingredients = 8;
3 let steps = 0;
4 // PJsMaker, simulateError();
5
6
7 function fetchIngredients() {
8   PJsMaker.fetchIngredients();
9   PJsMaker.fetchIngredients();
10  }
11
12 function handleIngredients(err) {
13   if (err) return console.log("An error occurred while fetching the ingredients");
14   ingredients++;
15   if (ingredients === 3) {
16     prepareSandwich();
17   }
18 }
19
20 function prepareSandwich() {
21   PJsMaker.spreadPeasAndButter();
22   PJsMaker.spreadJams();
23 }
24
25 function preparationStepCompleted(err) {
26   if (err) return console.log("An error occurred while preparing the sandwich");
27   steps++;
28   if (steps === 3) {
29     completeSandwich();
30   }
31 }
32
33 function completeSandwich(err, result) {
34   PJsMaker.closeSandwich();
35   if (err) {
36     console.log("An error occurred");
37   } else {
38     console.log("Eat that " + result);
39   }
40 }
41
42 fetchIngredients();
```

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

Code

```
1 let PbnMaker = require('./pbnmaker');
2 let ingredients = 8;
3 let steps = 0;
4 // PBnMaker.prototype.fetch();
5
6 function fetchIngredients() {
7   pbnMaker.fetchIngredients();
8   PbnMaker.fetchPeanutButter(fetchIngredients);
9   PbnMaker.fetchJam(fetchIngredients);
10 }
11
12 function handleIngredients(err) {
13   if (err) return console.log("An error occurred while fetching the ingredients");
14   ingredients--;
15   if (ingredients === 0) {
16     prepareSandwich();
17   }
18 }
19
20 function prepareSandwich() {
21   PbnMaker.spreadPeanutButter(preparationStepCompleted);
22   PbnMaker.spreadJam(preparationStepCompleted);
23 }
24
25 function preparationStepCompleted(err) {
26   if (err) return console.log("An error occurred while preparing the sandwich");
27   steps++;
28   if (steps === 2) {
29     completeSandwich();
30   }
31 }
32
33 function completeSandwich() {
34   PbnMaker.completeSandwich(err, result) -- {
35     if (err) {
36       console.log("An error occurred");
37     } else {
38       console.log("Eat that " + result);
39     }
40   };
41 }
42
43 fetchIngredients();
```

```
function prepareSandwich() {
```

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



@joel__lord
#iJS18

Code

```
1 let PJsMaker = require('./pjsmaker');
2 let ingredients = 8;
3 let steps = 0;
4 // ER:lower, simulateError();
5
6 function fetchIngredients() {
7   PJsMaker.fetchIngredients();
8   PJsMaker.fetchIngredients();
9   PJsMaker.fetchIngredients();
10 }
11
12 function handleIngredients(err) {
13   if (err) return console.log("An error occurred while fetching the ingredients");
14   ingredients++;
15   if (ingredients === 3) {
16     prepareSandwich();
17   }
18 }
19
20 function prepareSandwich() {
21   PJsMaker.spreadPeasAndButter();
22   PJsMaker.spreadJam(preparationStepCompleted);
23 }
24
25 function preparationStepCompleted(err) {
26   if (err) steps = console.log("An error occurred while preparing the sandwich");
27   steps++;
28   if (steps === 2) {
29     completeSandwich();
30   }
31 }
32
33 function completeSandwich(err, result) {
34   PJsMaker.closeSandwich(err, result) -- {
35     if (err) {
36       console.log("An error occurred");
37     } else {
38       console.log("Eat that " + result);
39     }
40   };
41 }
42
43 fetchIngredients();
```

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

Code

```
1 let PbnMaker = require('./pbnmaker');
2 let ingredients = 8;
3 let steps = 0;
4 // PBnMaker.closeSandwich();
5
6 function fetchIngredients() {
7   PbnMaker.fetchIngredients();
8   PbnMaker.fetchIngredients();
9   PbnMaker.fetchIngredients();
10 }
11
12 function handleIngredients(err) {
13   if (err) return console.log("An error occurred while fetching the ingredients");
14   ingredients++;
15   if (ingredients === 3) {
16     prepareSandwich();
17   }
18 }
19
20 function prepareSandwich() {
21   PbnMaker.spreadPeanutButter();
22   PbnMaker.spreadJams();
23 }
24
25 function preparationStepCompleted(err) {
26   if (err) return console.log("An error occurred while preparing the sandwich");
27   steps++;
28   if (steps === 3) {
29     completeSandwich();
30   }
31 }
32
33 function completeSandwich(err) {
34   PbnMaker.closeSandwich(err, result) => {
35     if (err) {
36       console.log("An error occurred");
37     } else {
38       console.log("Eat that " + result);
39     }
40   }
41 }
42
43 fetchIngredients();
```

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



@joel__lord
#iJS18

Code

```
1 let PJsMaker = require('./pjsmaker');
2 let ingredients = 8;
3 let steps = 0;
4 // PJsMaker, simulateError();
5
6 function fetchIngredients() {
7   PJsMaker.fetchIngredients();
8   PJsMaker.fetchPeanutButter(fetchIngredients);
9   PJsMaker.fetchJam(fetchIngredients);
10 }
11
12 function handleIngredients(err) {
13   if (err) return console.log("An error occurred while fetching the ingredients");
14   ingredients++;
15   if (ingredients === 3) {
16     prepareSandwich();
17   }
18 }
19
20 function prepareSandwich() {
21   PJsMaker.spreadPeanutButter(preparationStepCompleted);
22   PJsMaker.spreadJam(preparationStepCompleted);
23 }
24
25 function preparationStepCompleted(err) {
26   if (err) console.log("An error occurred while preparing the sandwich");
27   steps++;
28   if (steps === 2) {
29     completeSandwich();
30   }
31 }
32
33 function completeSandwich() {
34   PJsMaker.closeSandwich(err, result) => {
35     if (err) {
36       console.log("An error occurred");
37     } else {
38       console.log("Eat that " + result);
39     }
40   });
41 }
42
43 fetchIngredients();
```

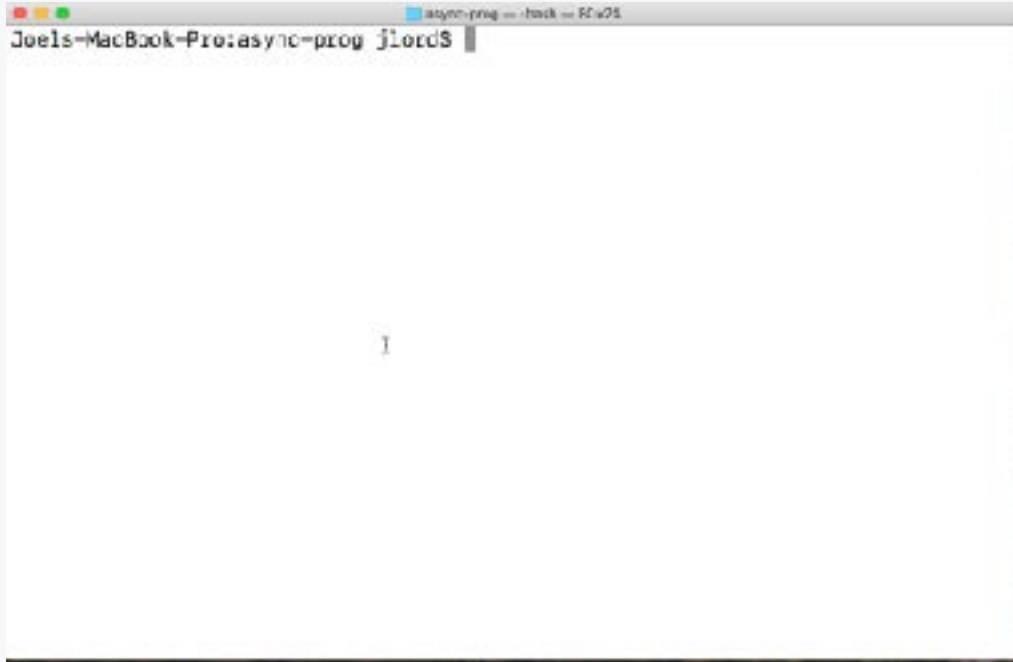
fetchIngredients();




@joel__lord
#iJS18

Result

Parallel



A screenshot of a terminal window. The title bar at the top reads "async-prog -- host -- macOS". The terminal prompt is "Joels-MacBook-Pro:async-prog jlord\$". The cursor is positioned at the end of the prompt line.



**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

Result

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

PBnJMaker.fetchBread()



@joel__lord
#iJS18

Code

Result

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

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



@joel__lord
#iJS18

Code

Result

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

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



@joel__lord
#iJS18

Code

Result

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

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


Code

Result

```
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 });
```

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

Code

Result

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

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



@joel__lord
#iJS18

Code

Result

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

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



@joel__lord
#iJS18

Code

Result

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

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



@joel__lord
#iJS18

Code

Result

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

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



@joel__lord
#iJS18

Code

```
1 let PBnJMaker = require("./pbnmaker");
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 });
```

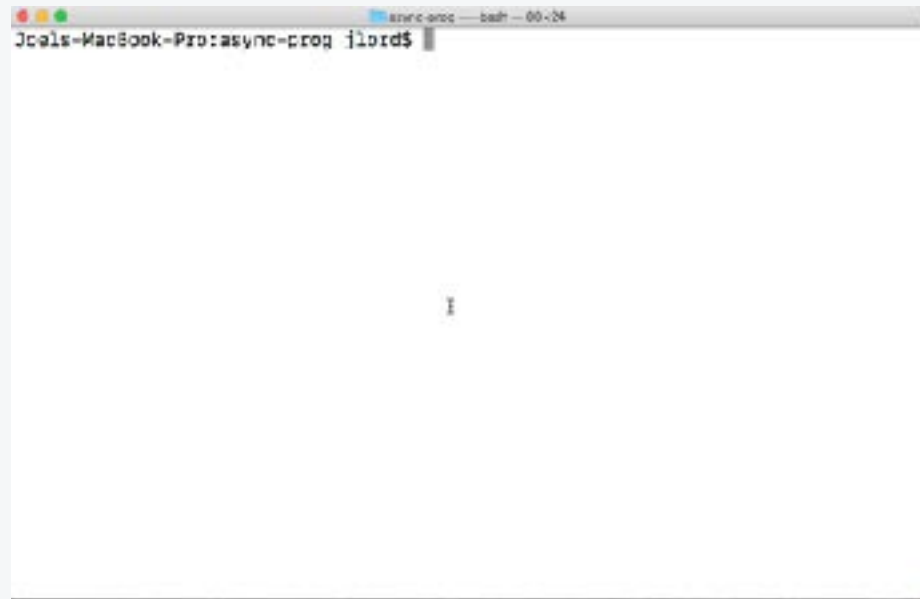
Result

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

Code

```
1 let PBnJMaker = require('./pbmjmaker');
2
3 PBnJMaker.simulateError();
4
5 PBnJMaker.fetchBread().then(() => {
6   return PBnJMaker.fetchPeanutButter();
7 }).then(() => {
8   return PBnJMaker.fetchJaw();
9 }).catch((err) => {
10  console.log('Error fetching the ingredients');
11 }).then(() => {
12  return PBnJMaker.spreadPeanutButter();
13 }).then(() => {
14  return PBnJMaker.spreadJaw();
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



Code

```
1 let PBnJMaker = require('./pbmjmaker');
2
3 PBnJMaker.simulateError();
4
5 PBnJMaker.fetchBread().then(() => {
6   return PBnJMaker.fetchPeanutButter();
7 }).then(() => {
8   return PBnJMaker.fetchJaw();
9 }).catch((err) => {
10  console.log('Error fetching the ingredients');
11 }).then(() => {
12  return PBnJMaker.spreadPeanutButter();
13 }).then(() => {
14  return PBnJMaker.spreadJaw();
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

```
1 let PBnJMaker = require('./pbnJmaker');
2
3 PBnJMaker.simulateError();
4
5 PBnJMaker.fetchBread().then(() => {
6   return PBnJMaker.fetchPeanutButter();
7 }).then(() => {
8   return PBnJMaker.fetchJag();
9 }).catch((err) => {
10  console.log("Error fetching the ingredients");
11 }).then(() => {
12  return PBnJMaker.spreadPeanutButter();
13 }).then(() => {
14  return PBnJMaker.spreadJag();
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");
```

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

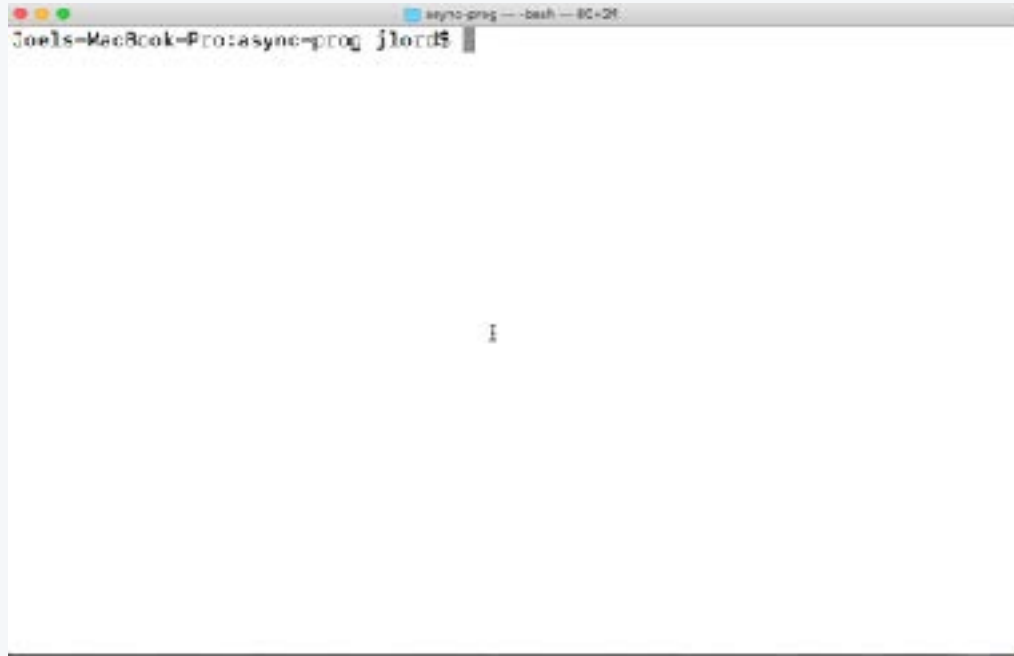
```
}).catch((err) => {
  console.log("Error somewhere");
```



@joel__lord
#iJS18

Result

Promises



A screenshot of a terminal window. The title bar shows "a1yno-prog -- bash -- 80x25". The terminal content shows the prompt "Joel's-MacBook-Pro: a1yno-prog jlord\$" followed by a cursor. The rest of the terminal is empty.

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);
  });
```

A white plate is centered on a blue and white checkered tablecloth. On the plate, there is a single slice of white bread at the top and a sandwich at the bottom. The sandwich is partially cut, revealing a filling of dark red jam and a light-colored spread. A black rectangular box is overlaid on the center of the plate, containing the text "Generators (2015)" in white.

**Generators
(2015)**

What is a generator?

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

```
function* iterable() {
  let i = 4;
  while(i-- > 0) {
    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-->0) {  
    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-->0) {  
    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-- > 0) {
    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-- > 0) {
    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-->0) {  
    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-->0) {  
    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-- > 0) {
    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-- > 0) {
    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-->0) {  
    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-->0) {  
    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-->0) {  
    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-->0) {  
    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-->0) {  
    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-->0) {  
    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-- > 0) {
    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-- > 0) {
    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-->0) {
    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-->0) {  
    yield i;  
  }  
}
```

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

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

Code

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

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
#iJS18

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

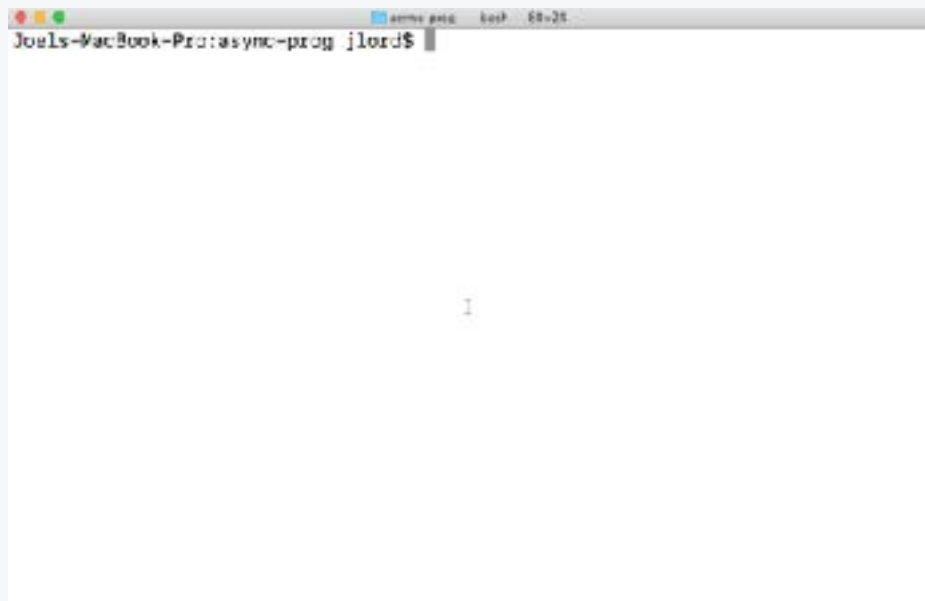
Code

```
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

Programmatically 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)
});
```



@joel__lord
#iJS18



**Events
(1990's)**

Events in the browser

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

```
1 <html>
2 <body>
3   <button id="btn1">Click me</button>
4 </body>
5
6 <script>
7   let btn = document.querySelector("#btn1");
8   btn.onclick = () => alert("Clicked");
9 </script>
10 </html>
```

Node Event Emitter

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

```
1 const EventEmitter = require('events');
2 class MyEmitter extends EventEmitter {}
3 const eventEmitter = new MyEmitter();
4
5 let fetchBread = function(options) {
6   return axios.get(SERVER + "/fetchBread").then((response) => {
7     eventEmitter.emit("breadFetched", response);
8   });
9 };
10
11 exports.Events = eventEmitter;
```

Node Event Emitter

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

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

Code

```
1 let PbnJMaker = require('./pbjmaker');
2
3 PbnJMaker.Events.on("breadFetched", () => {
4   PbnJMaker.fetchPeasutButter();
5 });
6 PbnJMaker.Events.on("peasutButterFetched", () => PbnJMaker.fetchJas());
7 PbnJMaker.Events.on("jasFetched", () => PbnJMaker.spreadPeasutButter());
8 PbnJMaker.Events.on("ocanwtButterSpreaded", () => PbnJMaker.spreadJam());
9 PbnJMaker.Events.on("jamSpreaded", () => PbnJMaker.cookSandwich());
10 PbnJMaker.Events.on("sandwichClosed", (result) => console.log("Eat that " + result));
11
12 PbnJMaker.fetchBread();
13
```

Demo



@joel__lord
#iJS18

Code


```
1 let FbnMaker = require('./fbMaker');
2
3 let ingredientsFetched = 0;
4 let ingredientsSpreaded = 0;
5
6 let ingredientFetched = () => {
7   ingredientsFetched++;
8   if (ingredientsFetched === 5) {
9     FbnMaker.Events.emit('IngredientFetched');
10  }
11 };
12
13 let spreaded = () => {
14   ingredientsSpreaded++;
15   if (ingredientsSpreaded === 2) {
16     FbnMaker.Events.emit('IngredientSpreaded');
17   }
18 };
19
20 FbnMaker.Events.on('breadFetched', ingredientFetched);
21 FbnMaker.Events.on('greenSBitterFetched', ingredientFetched);
22 FbnMaker.Events.on('jamFetched', ingredientFetched);
23
24 FbnMaker.Events.on('IngredientFetched', () => {
25   FbnMaker.spreadFennelEggs();
26   FbnMaker.spreadJam();
27 });
28
29 FbnMaker.Events.on('greenSBitterSpreaded', spreaded);
30 FbnMaker.Events.on('jamSpreaded', spreaded);
31
32 FbnMaker.Events.on('IngredientSpreaded', () => {
33   FbnMaker.closeSandwich();
34 });
35
36 FbnMaker.Events.on('sandwichClosed', result => console.log('Eat that ' + result));
37
38 FbnMaker.fetchBread();
39 FbnMaker.fetchGreenSBitter();
40 FbnMaker.fetchJam();
41
```

Demo

```
async-prog — bash — 80x24
CR2VVAAC-ITD5:async-prog jlord@e1e4r
```



@joel__lord
#JS18



Reactive Stream (2017)

RxJS

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

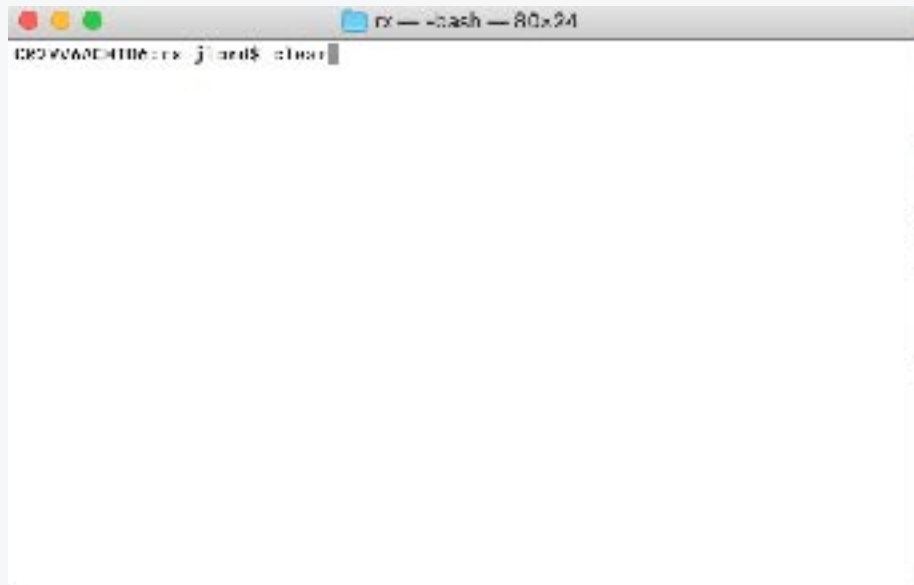
```
1 getDataFromLocalMemory()  
2   .filter (s => s !== null)  
3   .map(s => `${s} transformed`)  
4   .forEach(s => console.log(`next => ${s}`))
```

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

Code

```
1 const Rx = require("rxjs");
2 let PBnMaker = require("../pbmaker");
3
4 const ingredients = ["bread", "peanutButter", "jam"];
5
6 const ingredientEvents = ingredients.map((ingredient) => [
7   return Rx.Observable.fromEvent(PBnMaker.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 });
```

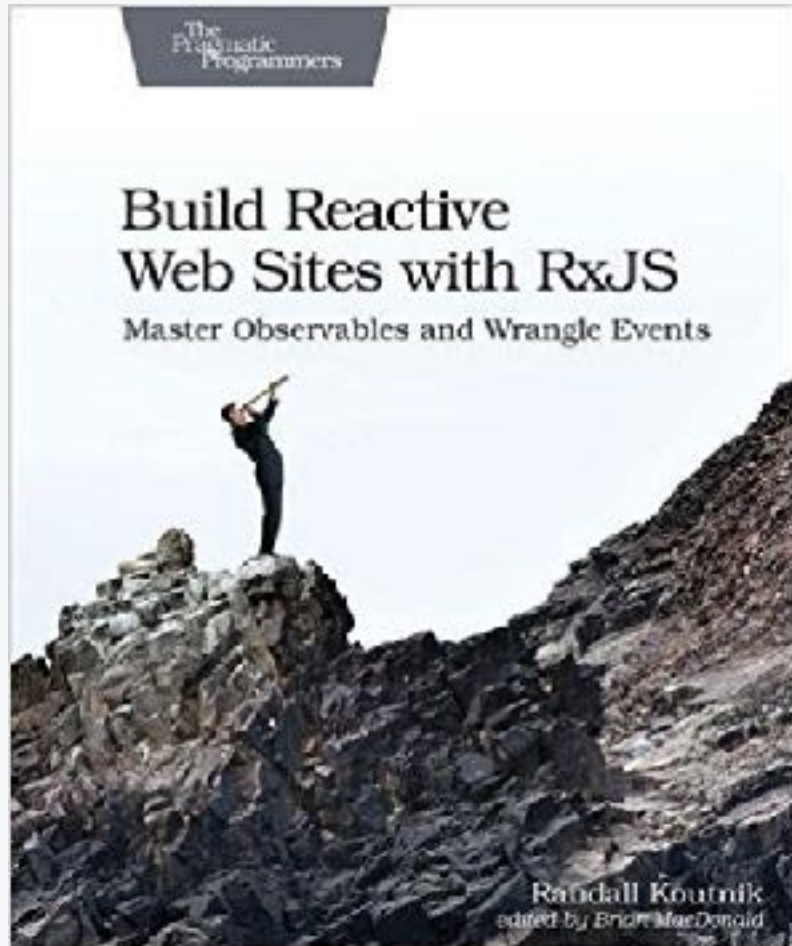
Demo




@joel__lord
#iJS18

RxJS

- RTFM



A close-up photograph of a person's hands holding a sandwich. The sandwich is made with white bread and filled with a red jam spread and fresh raspberries. The person is wearing a dark blue sweater. In the background, a white plate with another sandwich is visible on a wooden table. A dark semi-transparent rectangular box is overlaid on the center of the image, containing white text.

**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