# Iterating with Async/Await

**Nathan Taylor**
SOLUTION ARCHITECT

@taylonr www.taylonr.com

# What's the point?

# Async/await is syntactic sugar

# Syntactic Sugar

Syntax within a programming language that is designed to make things easier to read or to express.

*https://en.wikipedia.org/wiki/Syntactic_sugar*

Good news!

# Two Keywords

async

await

# Asynchronous Functions

function.js

```
async function getNames(){
    return [];
}
```

fat.arrow.js

```
const getNames = async () => {
    return [];
}
```

Return value is wrapped in a promise

# Important Facts About await

Must be used inside of async

Only blocks current function

# Blocking Function

await.js

```
const getNames = async () => {
    await someFunc();
    doSomethingElse();
}

getNames();
getAddresses();
```

# Two Approaches: One Goal

async/await

Promises

# Awaiting a Call

# Getting Data

promise.js

```
axios.get("/orders/1")

.then(({data}) => {
    setText(JSON.stringify(data))

});
```

await.js

```
const {data} = await
    axios.get("/orders/1");

setText(JSON.stringify(data));
```

# Handling Errors with Async/Await

# Chaining Async/Await

xhr.js

```
xhr.onload = () => {
    let xhr2 = new XMLHttpRequest();
    xhr2.open("GET", `/addresses/${shippingAddress}`);

    xhr2.onload = () => {
      setText(`City: ${JSON.parse(xhr2.response).city}`);
    };

    xhr2.send();
  };

 xhr.send();
```

promise.js

```
axios.get("orders/1")
    .then(({data}) => {
        return  axios.get(`/addresses/${data.shippingAddress}`);
    })
    .then(({data}) => {
        setText(`City: ${data.city}`);
    })
```

await.js

```
const { data } = await axios.get("/orders/1");
const { data: address } = await axios.get(
   `addresses/${data.shippingAddress}`
 );
 setText(`City: ${JSON.stringify(address.city)}`);
```

How can I make a non-sequential call?

# Awaiting Concurrent Requests

# Awaiting Parallel Calls

# Summary

# Promise States

**Pending**

**Fulfilled**

**Rejected**

# A Promise Object

promise.js

```
let temp = new Promise();
```

# Executor Function

promise.js

```
let temp = new Promise((resolve, reject) => {

});
```

# Two Approaches: One Goal

async/await

Promises

# Thank you

@taylonr