JavaScript Promises and Async Programming
https://app.pluralsight.com/course-player?clipId=86e7d54e-a053-4d41-a7c6-9e38e49f7486

JavaScript is a single-threaded programming language which means only one thing can happen at a time. ... That's where asynchronous JavaScript comes into play. Using asynchronous JavaScript (such as callbacks, promises, and async/await), you can **perform long network requests** without blocking the main thread

synchronous code is executed in sequence – each statement waits for the previous statement to finish before executing. Asynchronous code doesn't have to wait – your program can continue to run. You do this to keep your site or app responsive, reducing waiting time for the user

Synchronous Code

In *synchronous* programs, if you have two lines of code (L1 followed by L2), then L2 cannot begin running until L1 has finished executing.

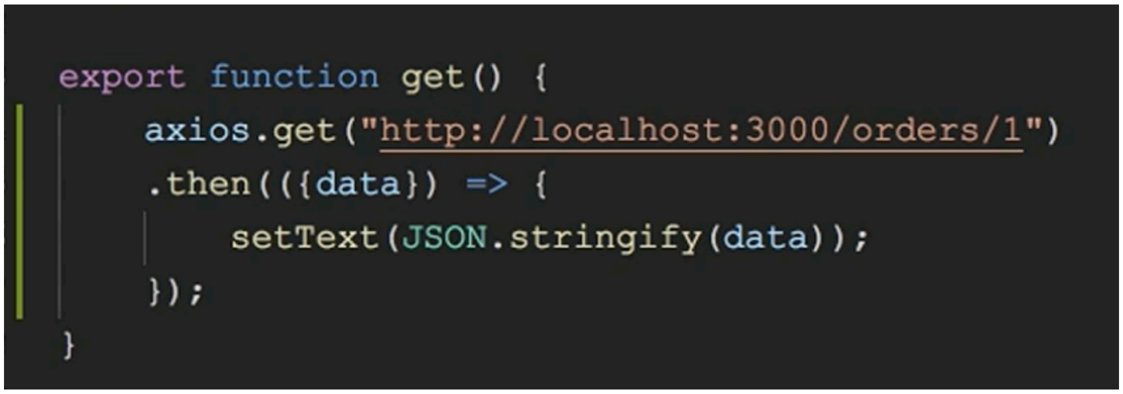
Asynchronous Code

In *asynchronous* programs, you can have two lines of code (L1 followed by L2), where L1 schedules some task to be run in the future, but L2 runs before that task completes.

Promises are **used to handle asynchronous operations in JavaScript**
axios is asynchronous library

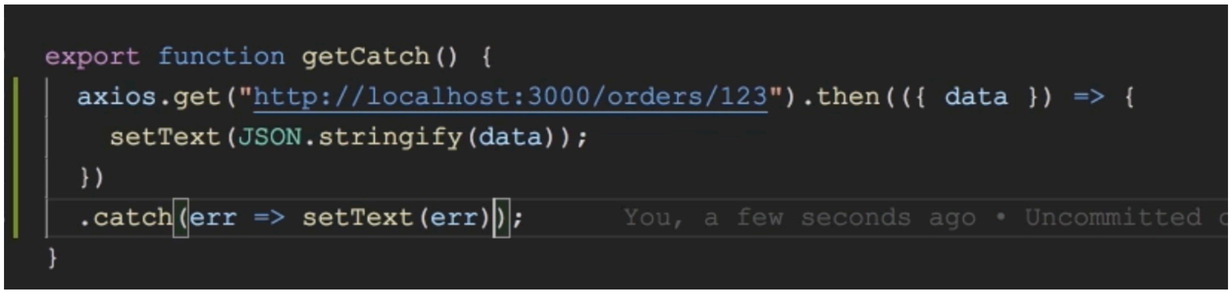
Promises - > Pending, Fulfilled, Rejected

If Promises are Fulfilled, called then()



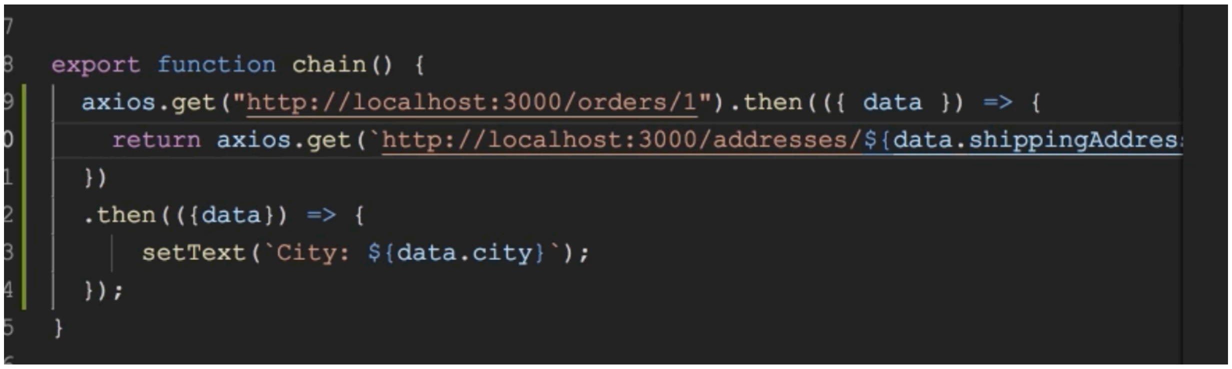
```
export function get() {
    axios.get("http://localhost:3000/orders/1")
    .then(({data}) => {
        setText(JSON.stringify(data));
    });
}
```

If Promises are Rejected, called catch()



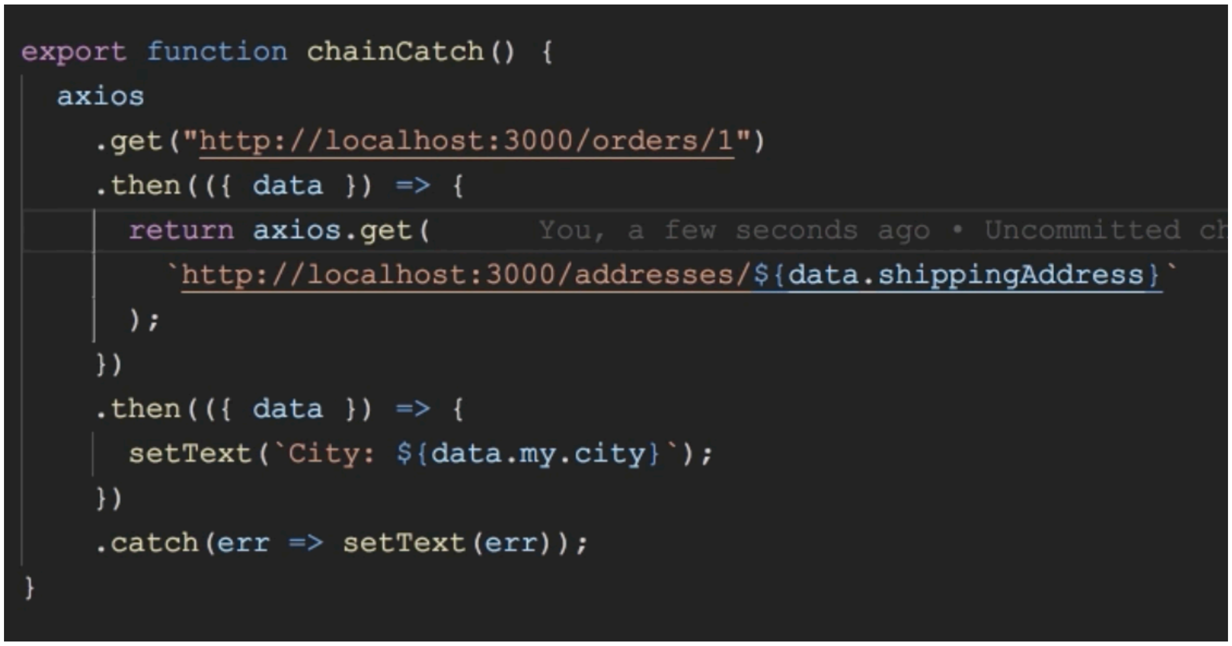
```
export function getCatch() {
  axios.get("http://localhost:3000/orders/123").then(({ data }) => {
    setText(JSON.stringify(data));
  })
  .catch(err => setText(err));       You, a few seconds ago • Uncommitted c
}
```

Multiple API calls with then() - One API depends on another API so we need to pass output to 2nd api from 1st api



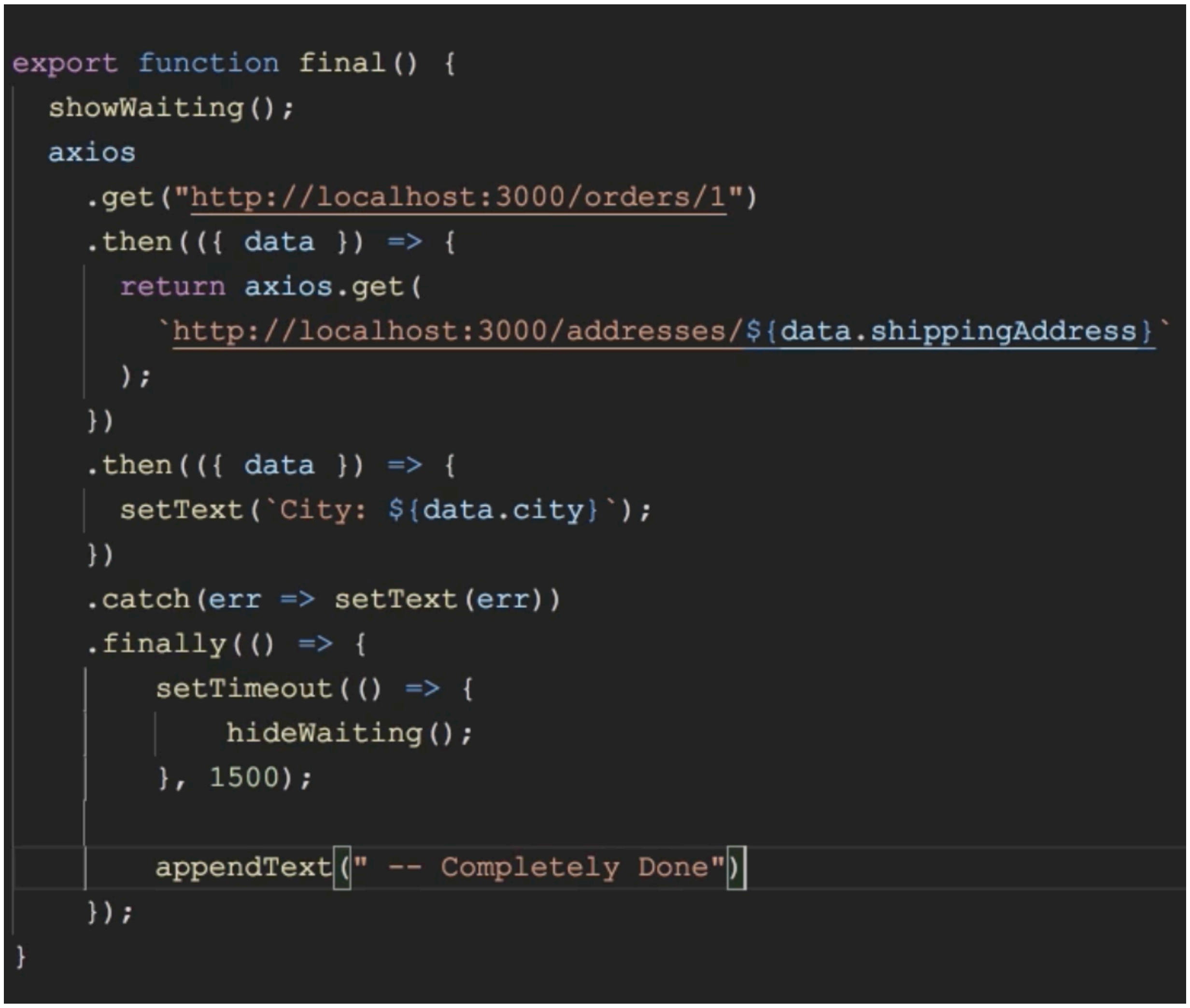
```
export function chain() {
  axios.get("http://localhost:3000/orders/1").then(({ data }) => {
    return axios.get(`http://localhost:3000/addresses/${data.shippingAddres
  })
  .then(({data}) => {
    setText(`City: ${data.city}`);
  });
}
```

Multiple API calls with then() & catch() - One API depends on another API so we need to pass output to 2nd api from 1st api



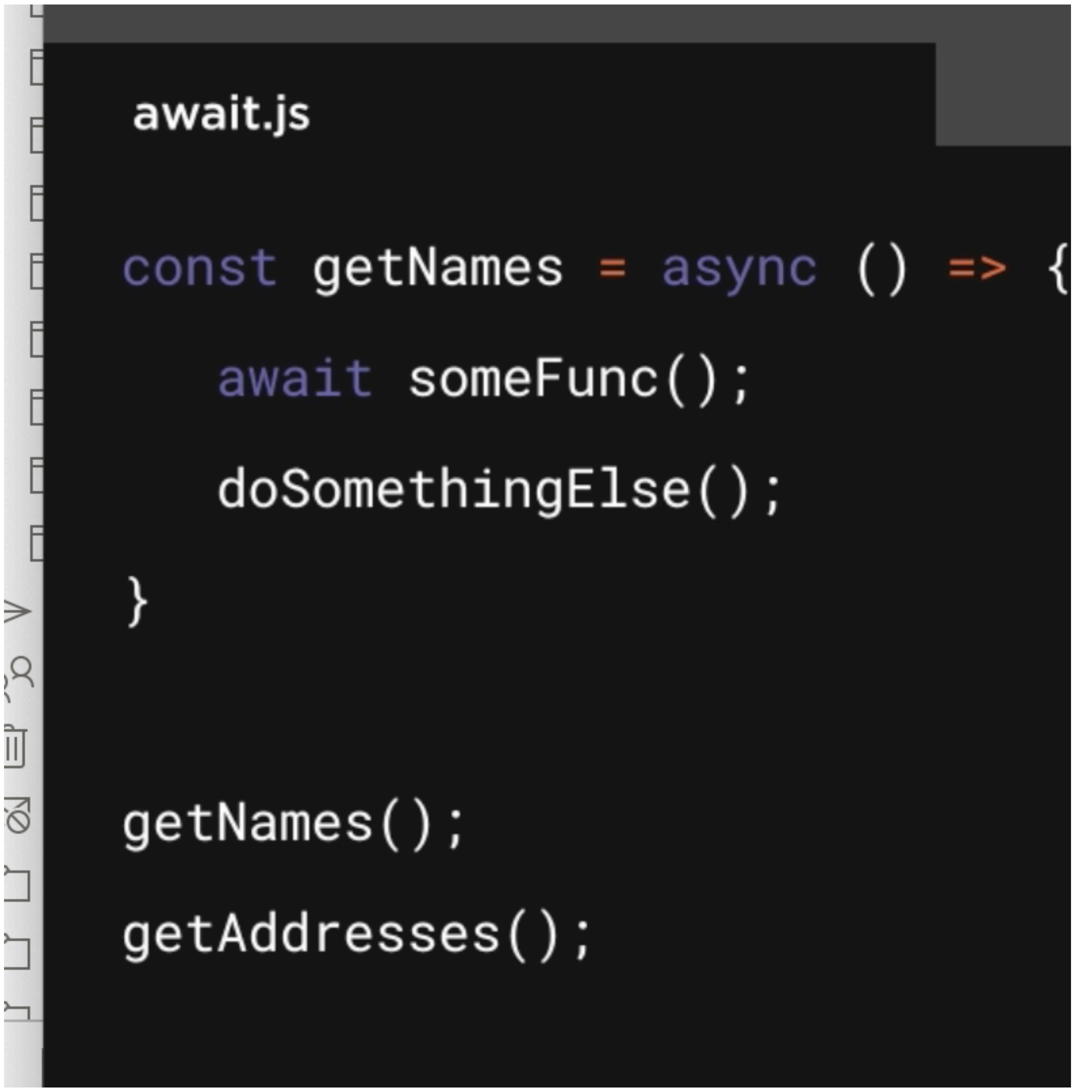
```
export function chainCatch() {
  axios
    .get("http://localhost:3000/orders/1")
    .then(({ data }) => {
      return axios.get(      You, a few seconds ago • Uncommitted ch
        `http://localhost:3000/addresses/${data.shippingAddress}`
      );
    })
    .then(({ data }) => {
      setText(`City: ${data.my.city}`);
    })
    .catch(err => setText(err));
}
```

Once Promises are settled, finally () will call irrespective of rejected / fulfilled



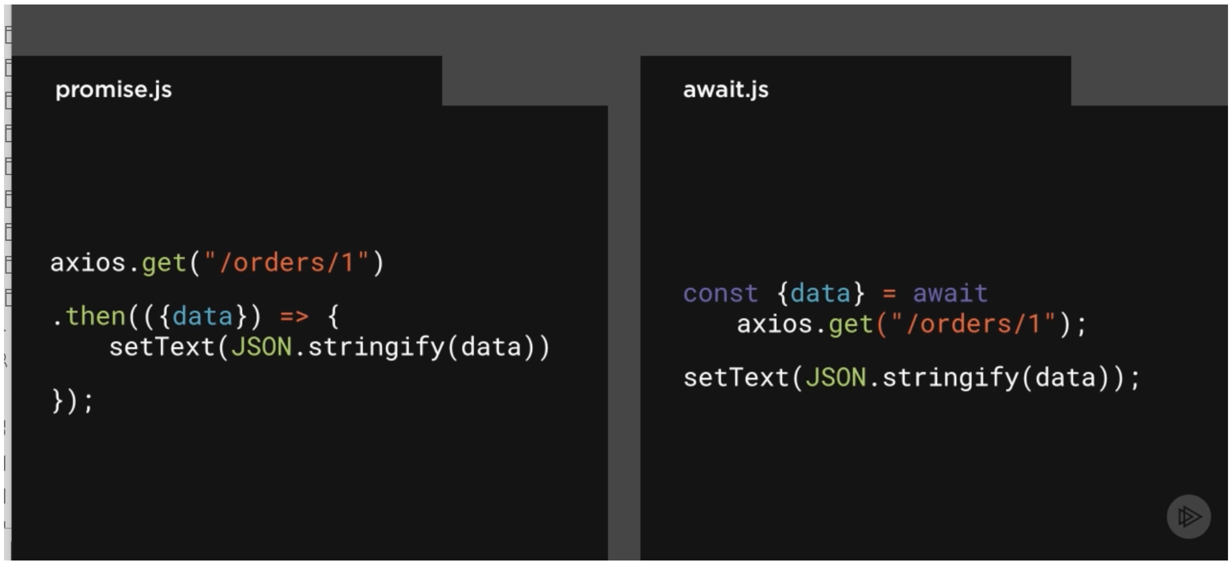
```
export function final() {
  showWaiting();
  axios
    .get("http://localhost:3000/orders/1")
    .then(({ data }) => {
      return axios.get(
        `http://localhost:3000/addresses/${data.shippingAddress}`
      );
    })
    .then(({ data }) => {
      setText(`City: ${data.city}`);
    })
    .catch(err => setText(err))
    .finally(() => {
        setTimeout(() => {
            hideWaiting();
        }, 1500);

        appendText(" -- Completely Done")
    });
}
```

Async/Await : easier than promises
- Async will return promise & Await will wait unit promises are fulfilled or rejected
- Return value is wrapper in a promise
- await must be used inside of async
- Only blocks current functions
- “doSomethingElse()” is blocked until someFunc() is done but remaining getNames() & getAddresses() are unblocked (that executes irrespective of someFunc() is done or not)



await.js

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

getNames();
getAddresses();
```

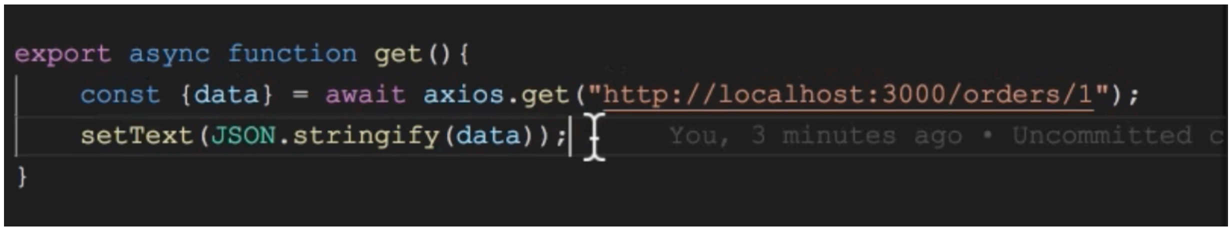


promise.js

```
axios.get("/orders/1")
.then(({data}) => {
    setText(JSON.stringify(data))
});
```

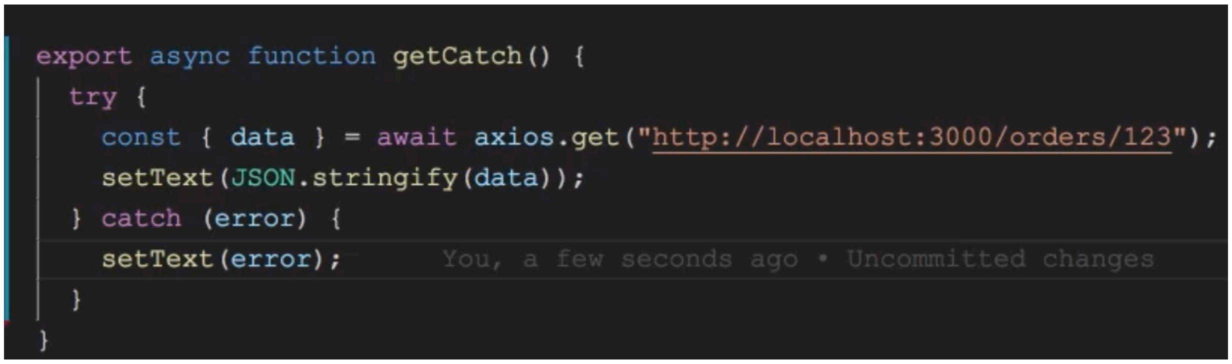
await.js

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



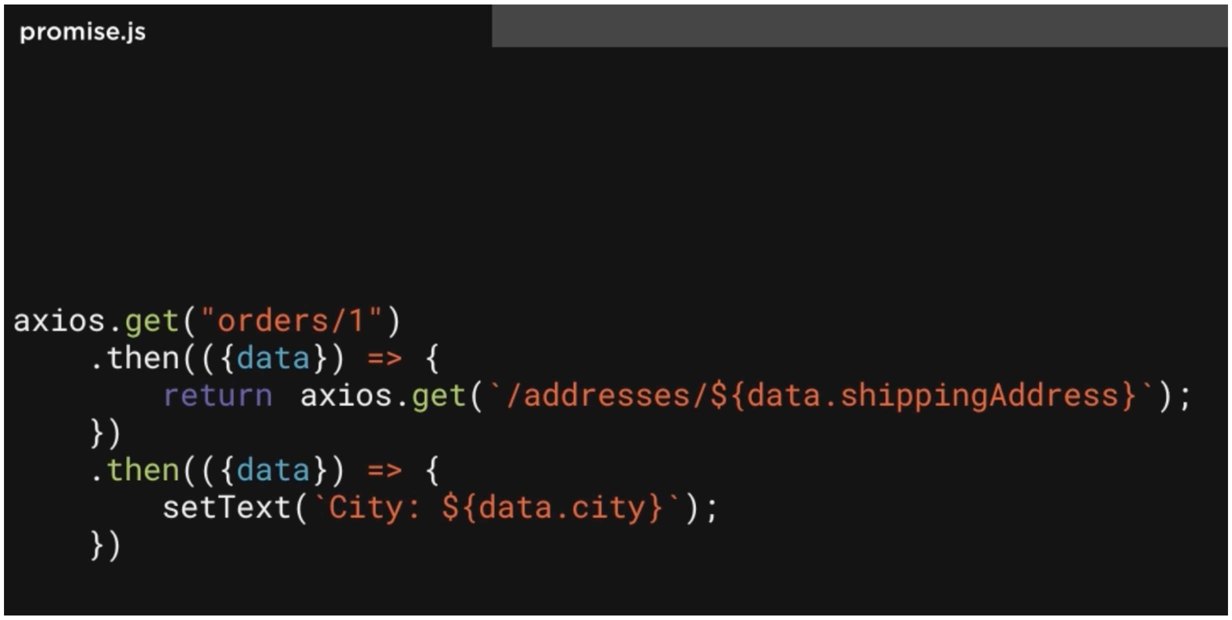
```
export async function get(){
    const {data} = await axios.get("http://localhost:3000/orders/1");
    setText(JSON.stringify(data));       You, 3 minutes ago • Uncommitted c
}
```

Error Handling

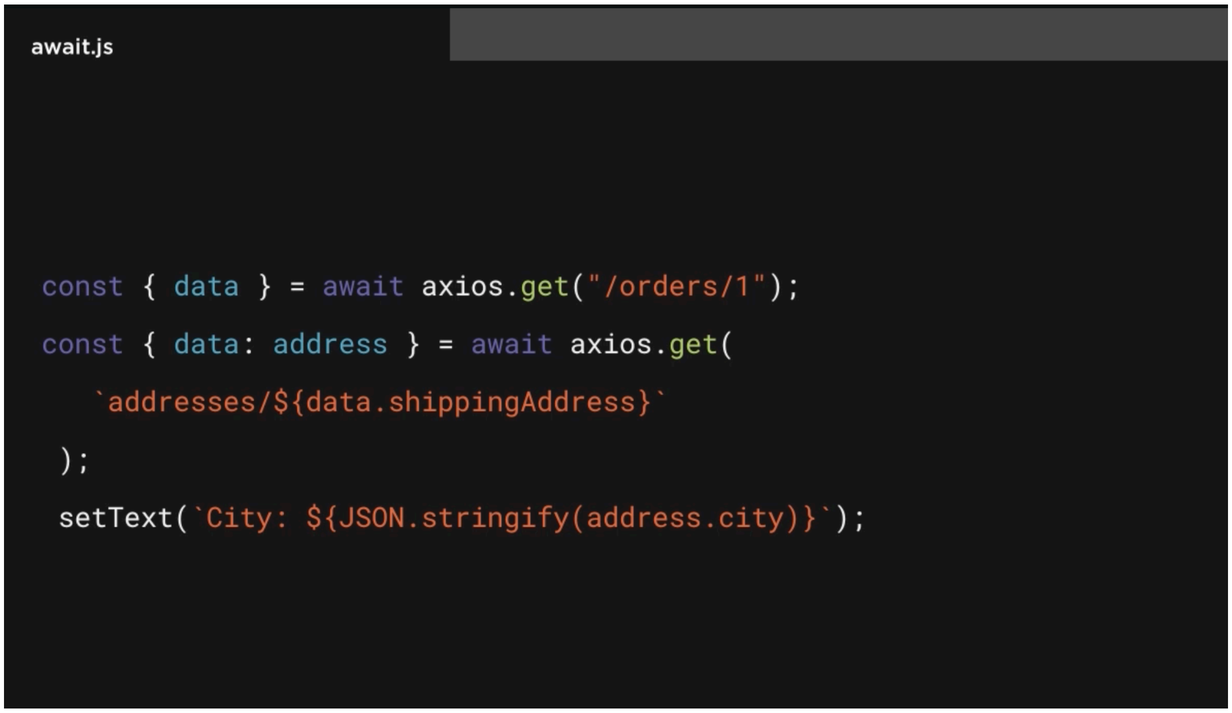


```
export async function getCatch() {
  try {
    const { data } = await axios.get("http://localhost:3000/orders/123");
    setText(JSON.stringify(data));
  } catch (error) {
    setText(error);       You, a few seconds ago • Uncommitted changes
  }
}
```

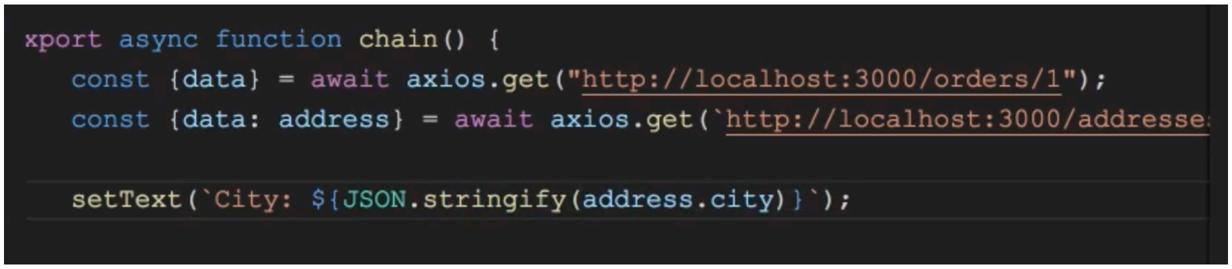
Chain (multiple API calls) - Sequential calls - depends API calls



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

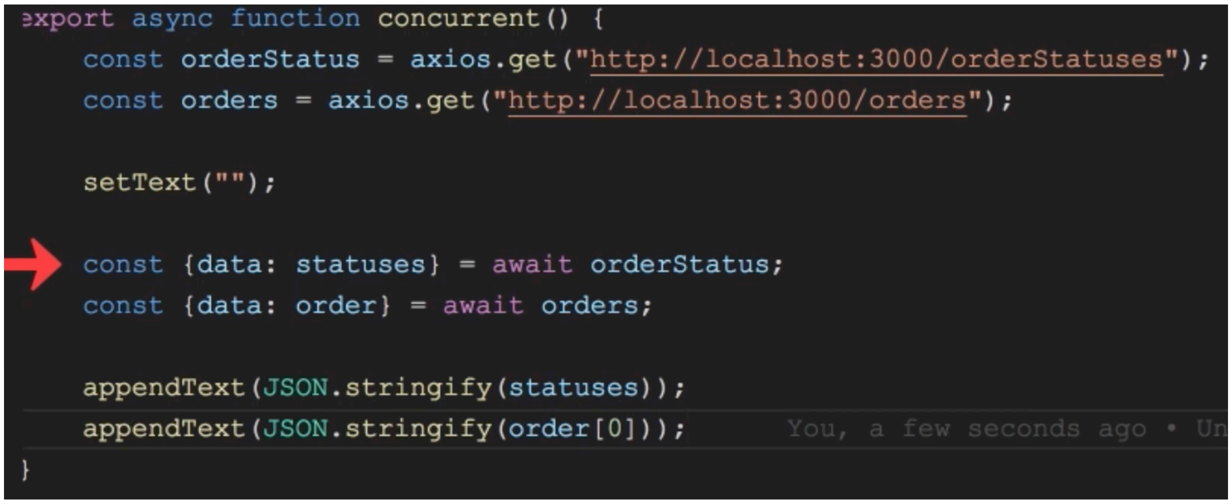

await.js

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



```
xport async function chain() {
   const {data} = await axios.get("http://localhost:3000/orders/1");
   const {data: address} = await axios.get(`http://localhost:3000/addresse

   setText(`City: ${JSON.stringify(address.city)}`);
```

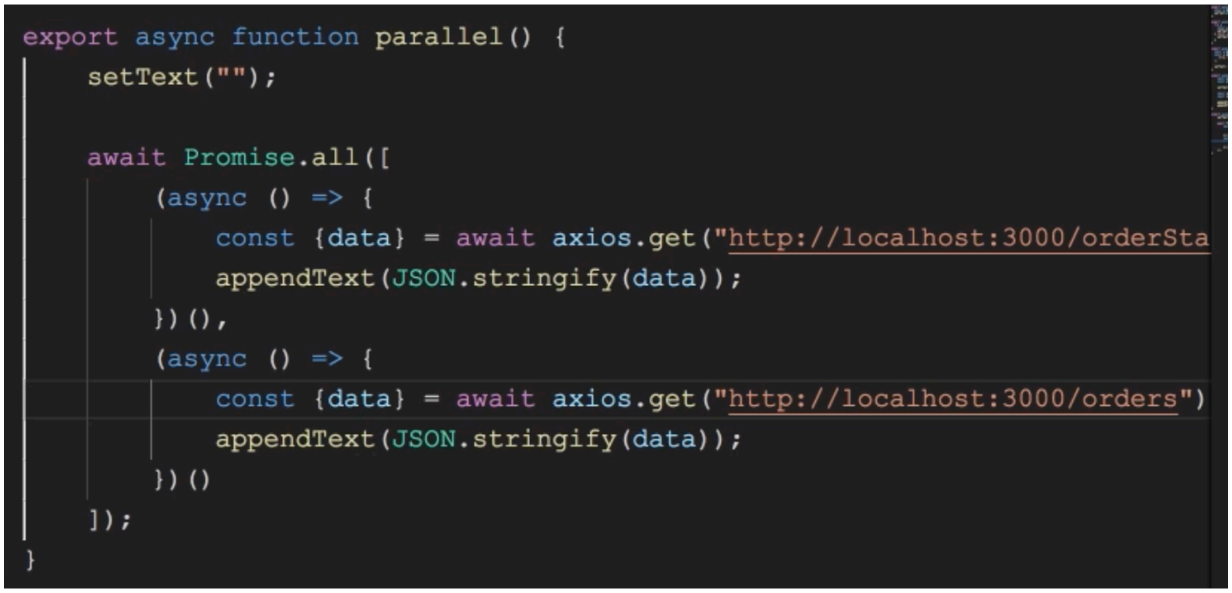
Chain (multiple API calls) - non - Sequential calls (concurrent calls) - we don't want API depends on other API's



```
export async function concurrent() {
    const orderStatus = axios.get("http://localhost:3000/orderStatuses");
    const orders = axios.get("http://localhost:3000/orders");

    setText("");

    const {data: statuses} = await orderStatus;
    const {data: order} = await orders;

    appendText(JSON.stringify(statuses));
    appendText(JSON.stringify(order[0]));       You, a few seconds ago • Un
}
```

Parallel API calls:

Promise.all will wait until all promises are done



```
export async function parallel() {
    setText("");

    await Promise.all([
        (async () => {
            const {data} = await axios.get("http://localhost:3000/orderSta
            appendText(JSON.stringify(data));
        })(),
        (async () => {
            const {data} = await axios.get("http://localhost:3000/orders")
            appendText(JSON.stringify(data));
        })()
    ]);
}
```