JavaScript Promises and Async Programming

Zawartość

[Promises 2](#_Toc69431189)

[Consume a Promise 5](#_Toc69431190)

[First Promise 5](#_Toc69431191)

[Handling with errors 6](#_Toc69431192)

[Chaining Promises Together 7](#_Toc69431193)

[Catch error in promise chain 7](#_Toc69431194)

[Creating and queuing promises 9](#_Toc69431195)

[Create promise and consideration about its state 9](#_Toc69431196)

[How to use promise 10](#_Toc69431197)

[Promises Queue 10](#_Toc69431198)

[Settling up Promises 12](#_Toc69431199)

[Async and await 14](#_Toc69431200)

[Introduce 14](#_Toc69431201)

[First async promise 14](#_Toc69431202)

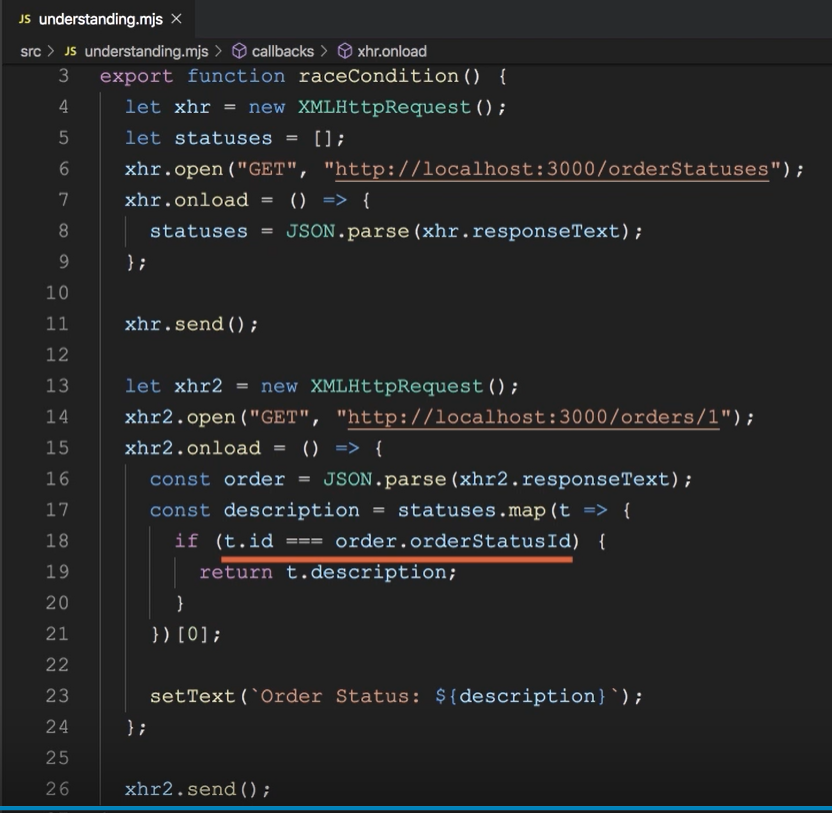
[Handling errors with async and await 14](#_Toc69431203)

[Chaining async/await 15](#_Toc69431204)

[Concurrent calls 15](#_Toc69431205)

[Parallel call 15](#_Toc69431206)

# Promises

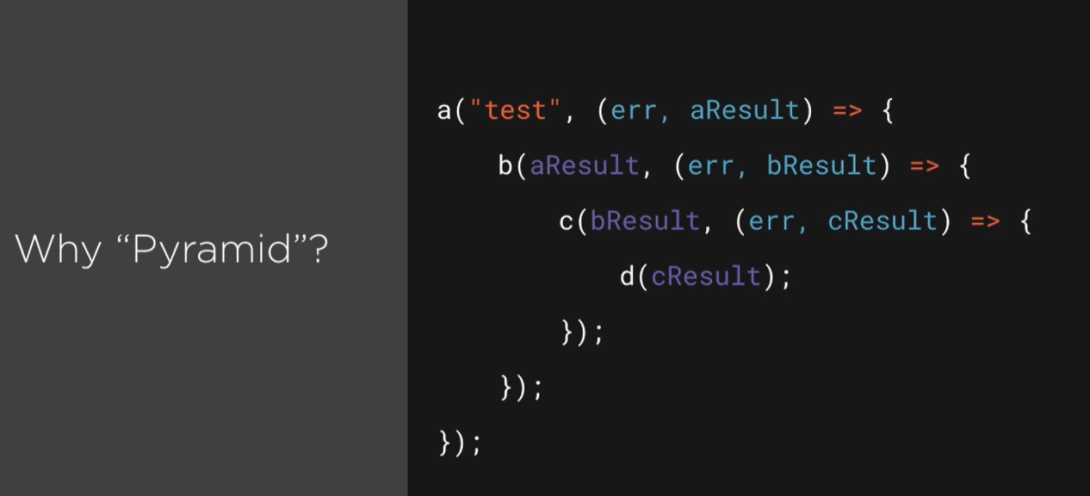


This code is not async. Naturally all action from async endpoint will be execute in futer, but in js code there is a if statement which check t.Id with orderStatusId. Enpoint is async and code will be execute in future so list of description is empty at this time.

XMLHttpRequest is an object which is use to make request for specific endpoint.

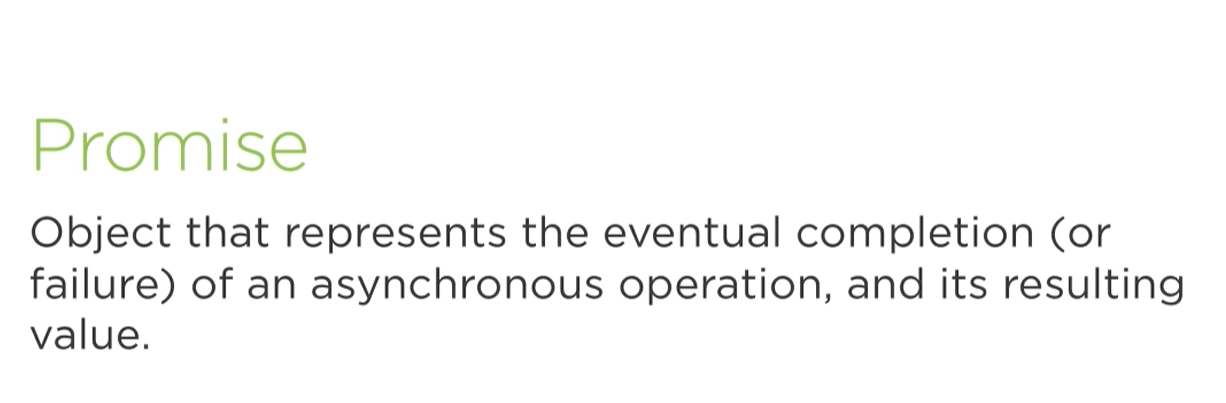


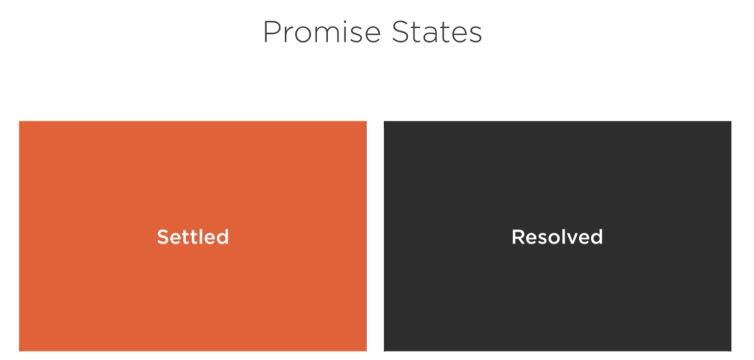
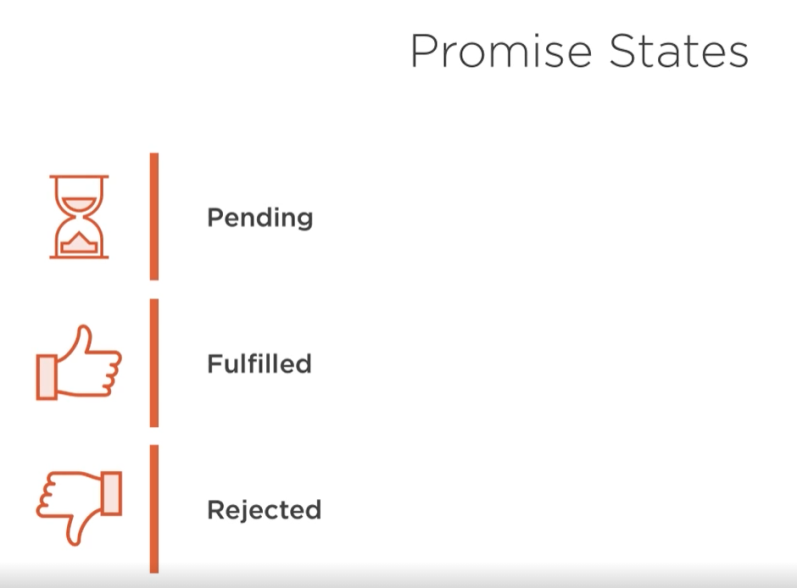
To make list with results. Invoke second Request inside first one. This method to copy with request is name pyramid callback.

****

There are a lot of nested callbacks it hard to read, maintain and tests.

Obviously we can solve this problem. Promise it is our solution.





This is promise which is each on rejected or fulfilled state.

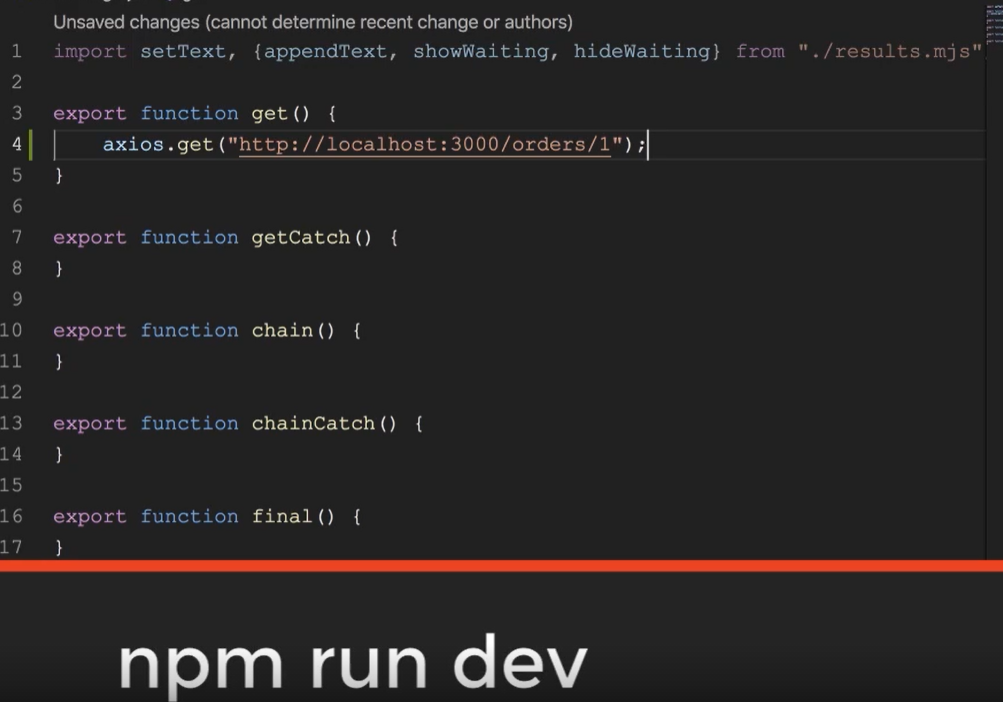


# Consume a Promise

## First Promise



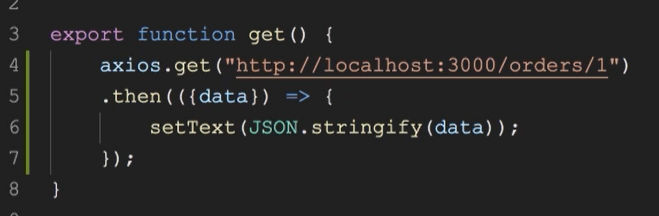
Library for promise



Npm run dev is a command to start a server to consume an API.



Be careful with create dbsets.

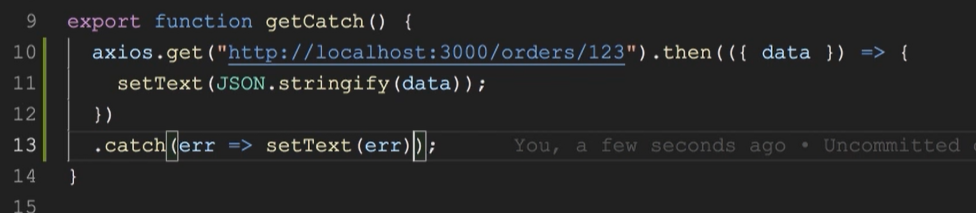


To display informations about http request use method then and map data to JSON type.

## Handling with errors



Error handling can be use in standard if else statement. If you pass result as param, you have access to status code in head and data from body.



To cope with errors use catch clause and do some action, which relate to error type.

## Chaining Promises Together

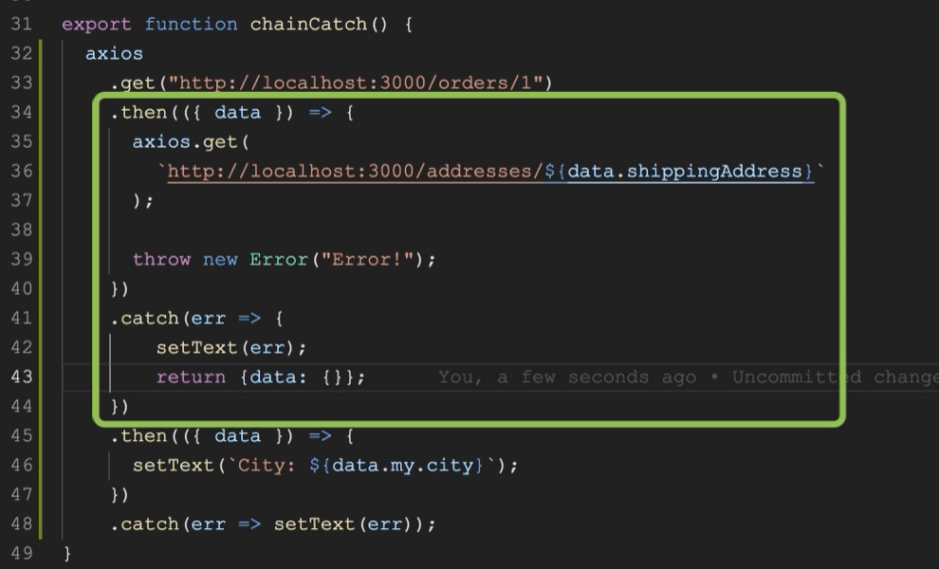


Promise chain is a promise which base on the first. In example there is a get request to orders/1 then in arrow function is invoke another promise base on the one before. Second promise return order/1 shipping address. Naturally to display data in html required is then method.

## Catch error in promise chain

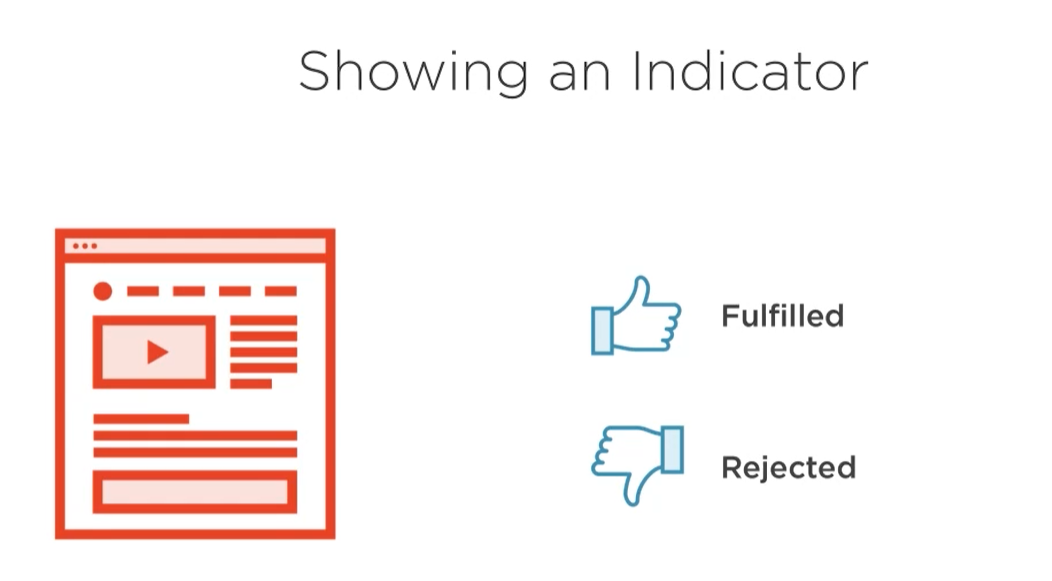


Chain promise works like pipeline. An error will be pass from each promise to last operation catch().

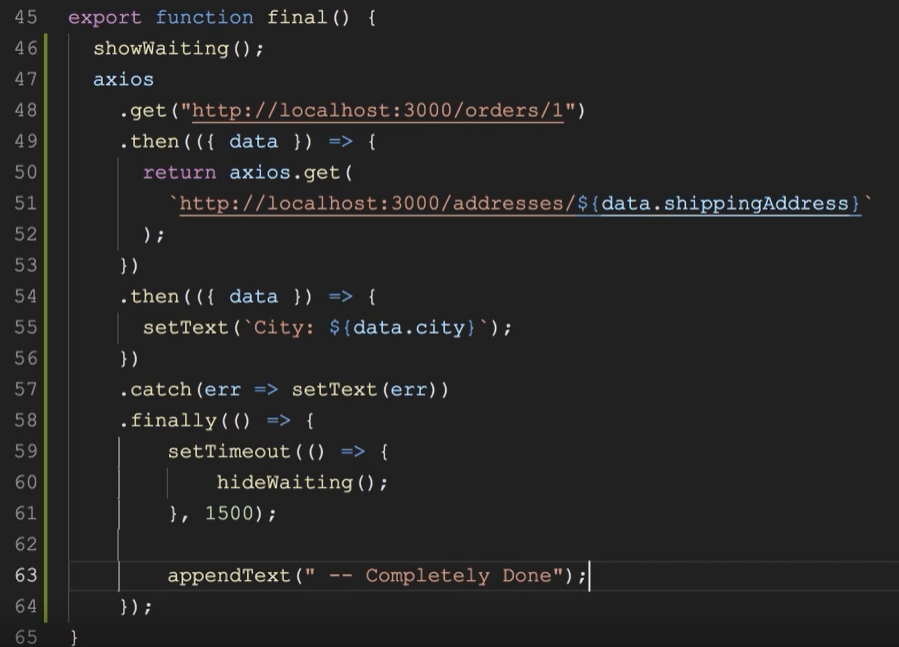


Catch method can be invoke after each then, but it relates only to each then before, not for all one. It is very intuitive because as I said before it works like pipeline. All errors occurred before catch can be display in catch method in this time.

Showing an Indicator.



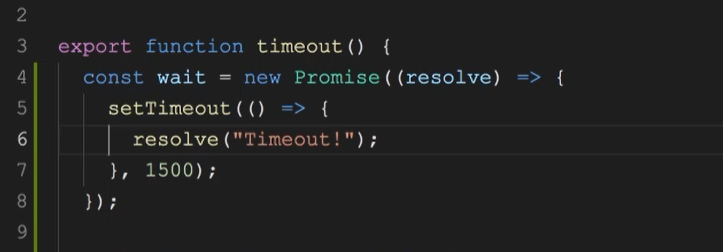
Indicator is a loader. It is use when promise is in pending state. Make it disappear when promise is fulfilled or rejected.



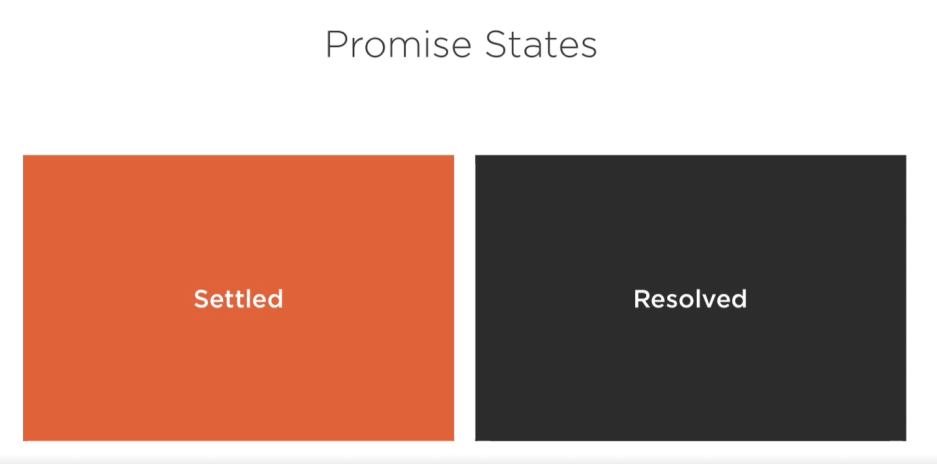
An example with loader.

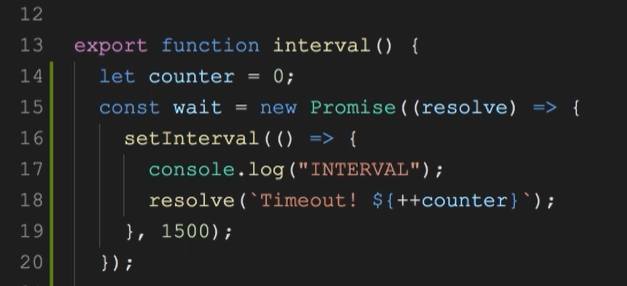
# Creating and queuing promises

## Create promise and consideration about its state



Each promise required executer function to change state from pending to fulfill or reject.





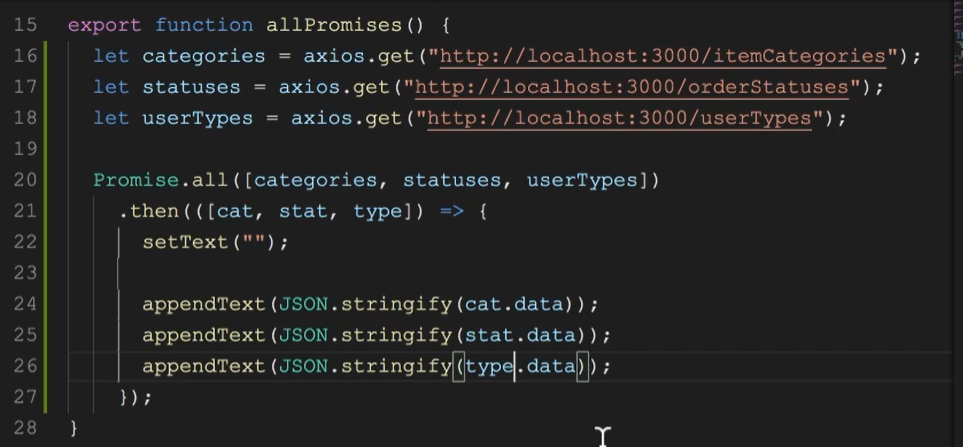
Let’s talk about two state of promise fulfillment. First one is settled. This status occur when promise param “resolve” passed into executor function do certain operations. It’s mean promise change state some object in our example changed has been a counter. This state of object name is settled. New state has been assigned to counter. However resolved status is when value was give by resolve function. Remember Settled, value assigned, resolved value was give.

## How to use promise

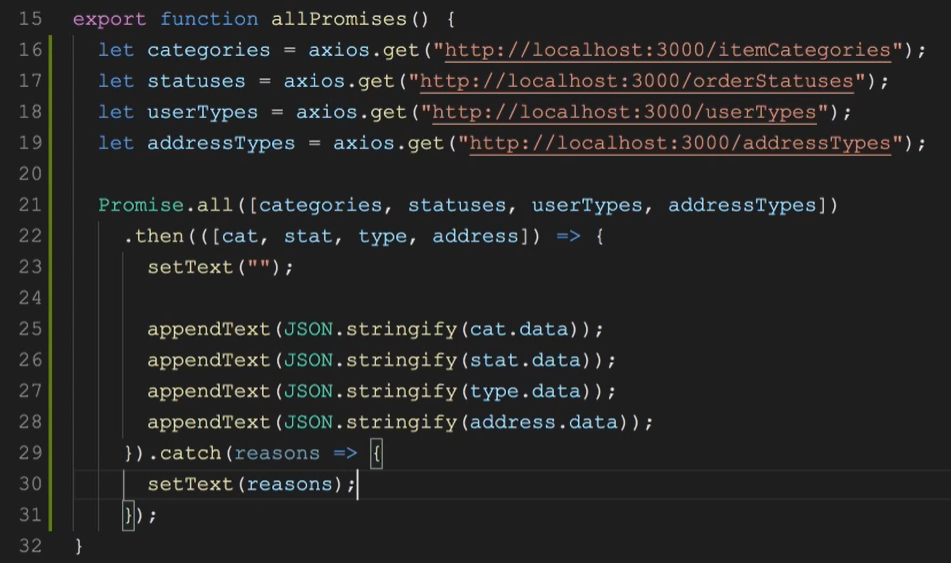


You met resolve param before. Now it’s time to talk about reject. If you mind it’s method for reject http request you are right. But to use this type of pram required is one thing more. It’s a status code. Status code is answer from server which is call in our request. Without if … else statement onerror method only show 5xx status code or if you prefer internal server problems. Otherwise with if … else clause can be assign (settled )each status code. Good practice is use if … else clause inside promise.

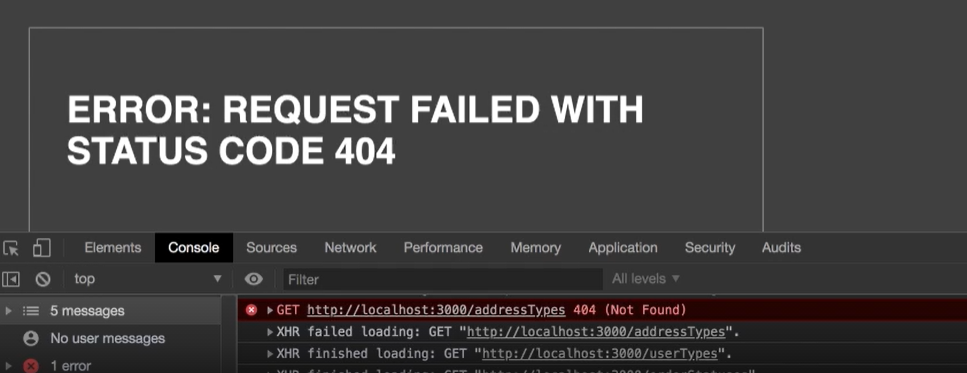
## Promises Queue



To be sure that all promises will be invoke in the same time required is Promise.all method as param this method array of promises. In this example used was axios instead of standard promise object with XHR object.



But there is another question what to do when one promise will return an error. In example above there is a new promise – addressType. When promises are in queue there are only two option each of them will be fulfill or one will be reject.



In error message is display an error. That means one from promises queue returned error code.

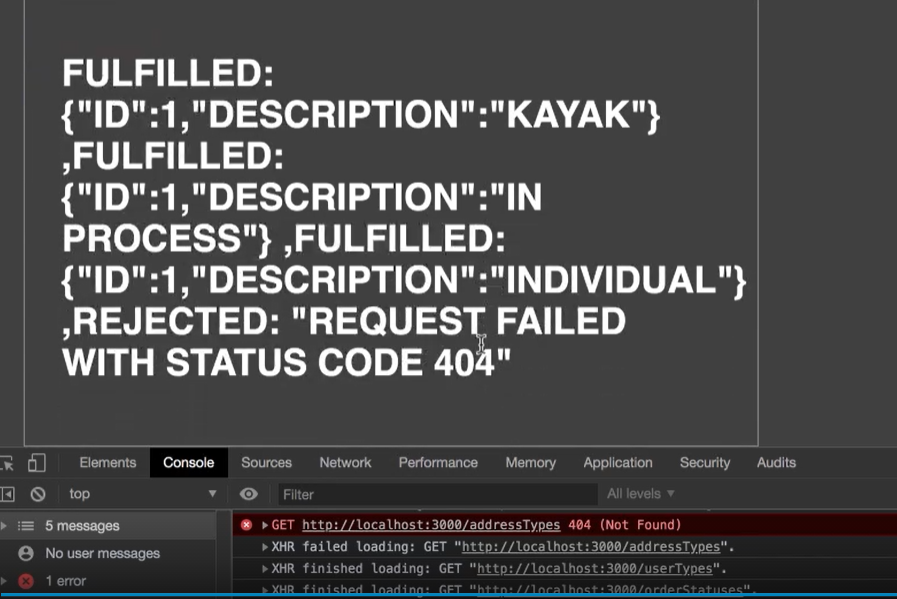
## Settling up Promises

Promises.all method describe in last lesson. It return resolve object as part of queue.

In this lessos we will talk about allSettled. First difference is returned data type.



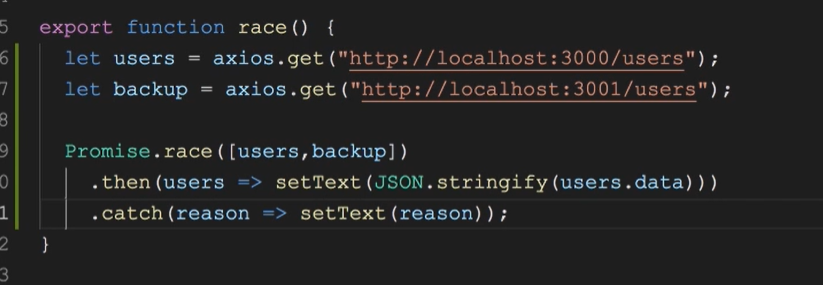
Now you see returned data is object with status name and two object. Resolve has value this is data form server. Rejected status has reason why this promises was reject by server. There is no need to use catch block to check reason but it’s still good practice.

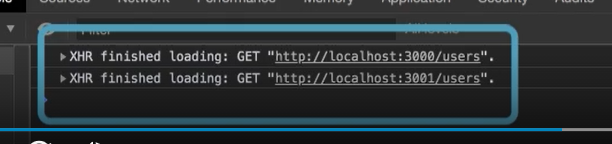


And there is a another difference, rule like in all method don’t exist (all fulfill or one rejected), each of promises return a result. Three of them an object and one reason why was rejected. But no all browsers support allSettled method.



The last option settled only the fastest promise.

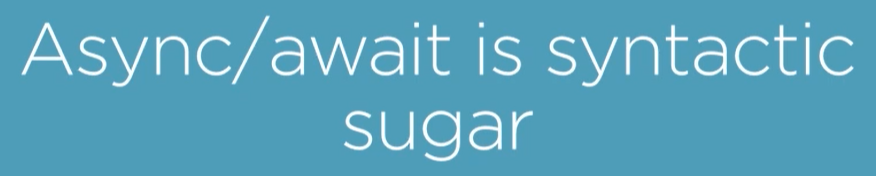


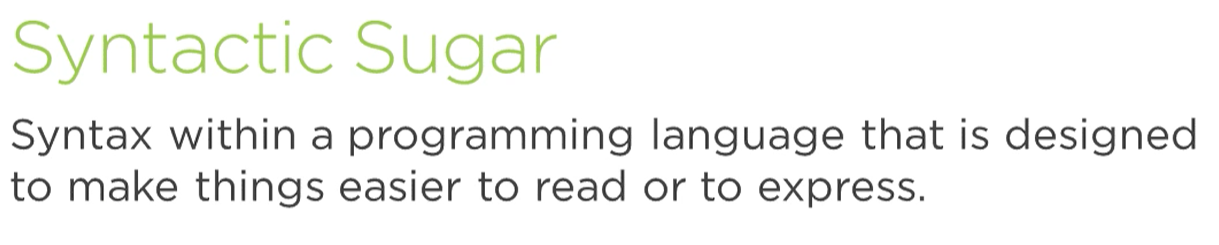


Data from localhost 3000 has been displayed, 3001 has been forgot.

# Async and await

## Introduce







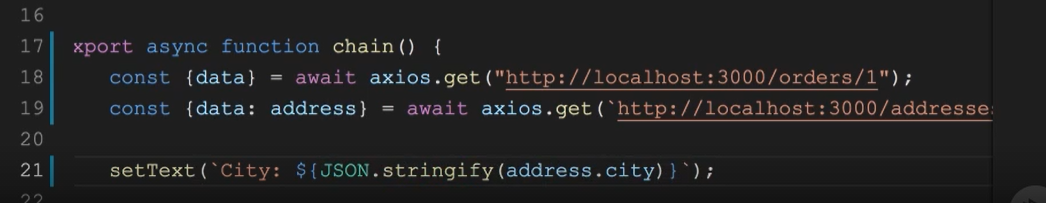
## First async promise



## Handling errors with async and await



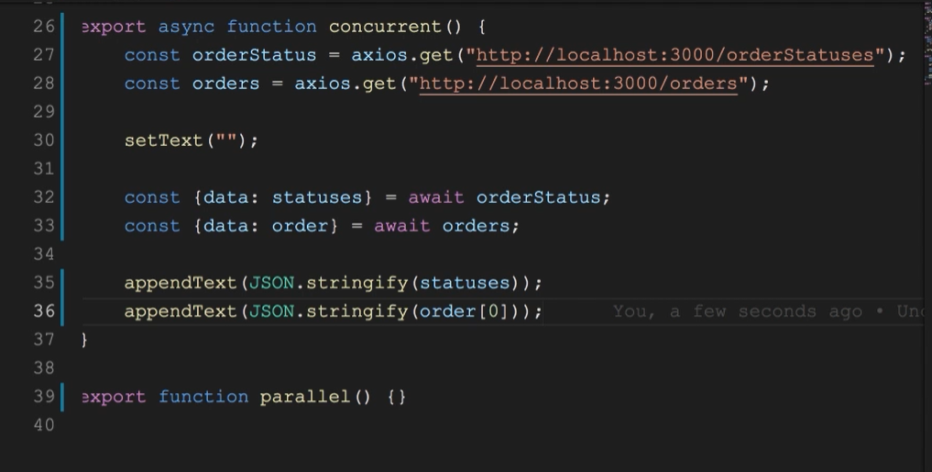
## Chaining async/await

****



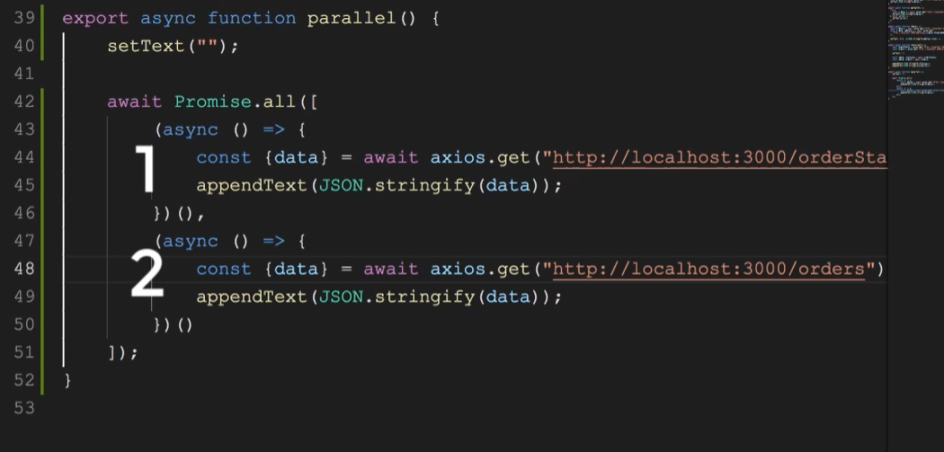
Only use data from first promise’s calls.

## Concurrent calls



In this demo, two calls has kicked of in the same time. Here in example important is await keyword. But first take a look on promises. When there are call there aren’t await keyword. This crucial part of async/await pair appears on 32nd and 33rd lines. That use of await make code concurrent. All promises started on the same time and finished almost in the same time.

## Parallel call



This approach is helpful to make few things in the same time. Each of request is calling in different thread.