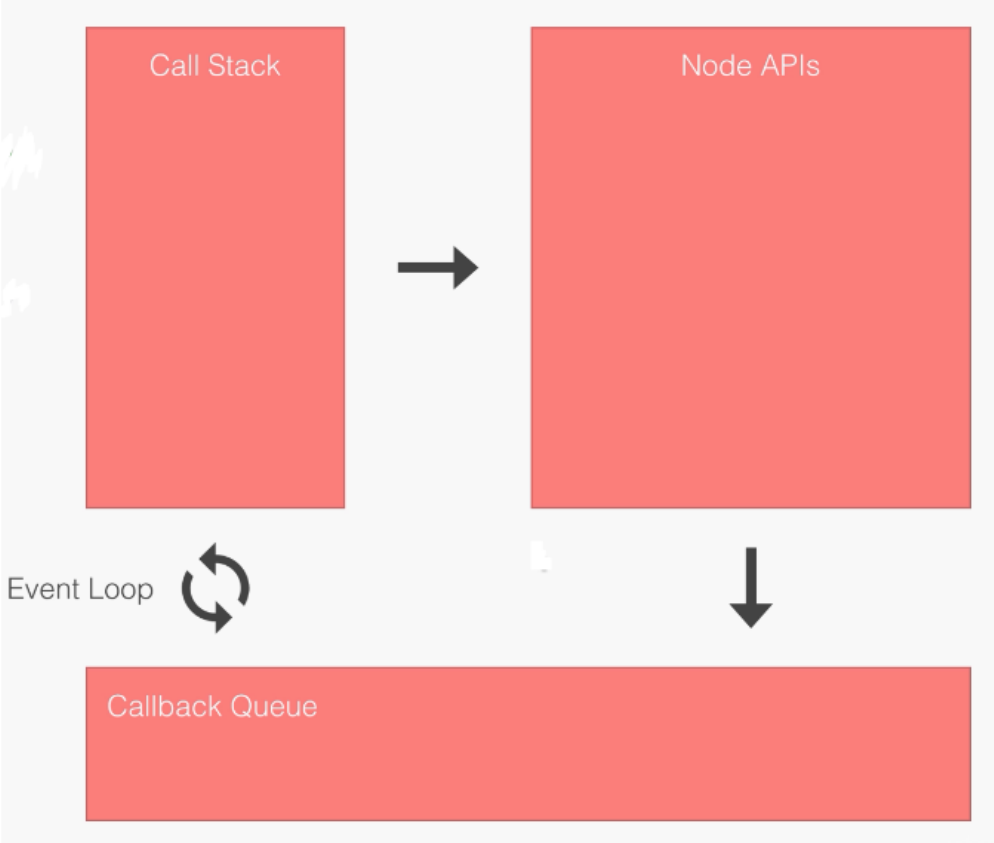
Asynchonous

ASYNCHRONOUS NODE.JS

What is it?

It's an Asynchronous execution of node js commands and functions.

How it work



#1: Call Stack:

Where all the **synchronous** code put in stack to the execution process.

#2: Node API:

Where all the **Asynchronous** code during execution process, then when they finish they move to:

#3: Callback Queue:

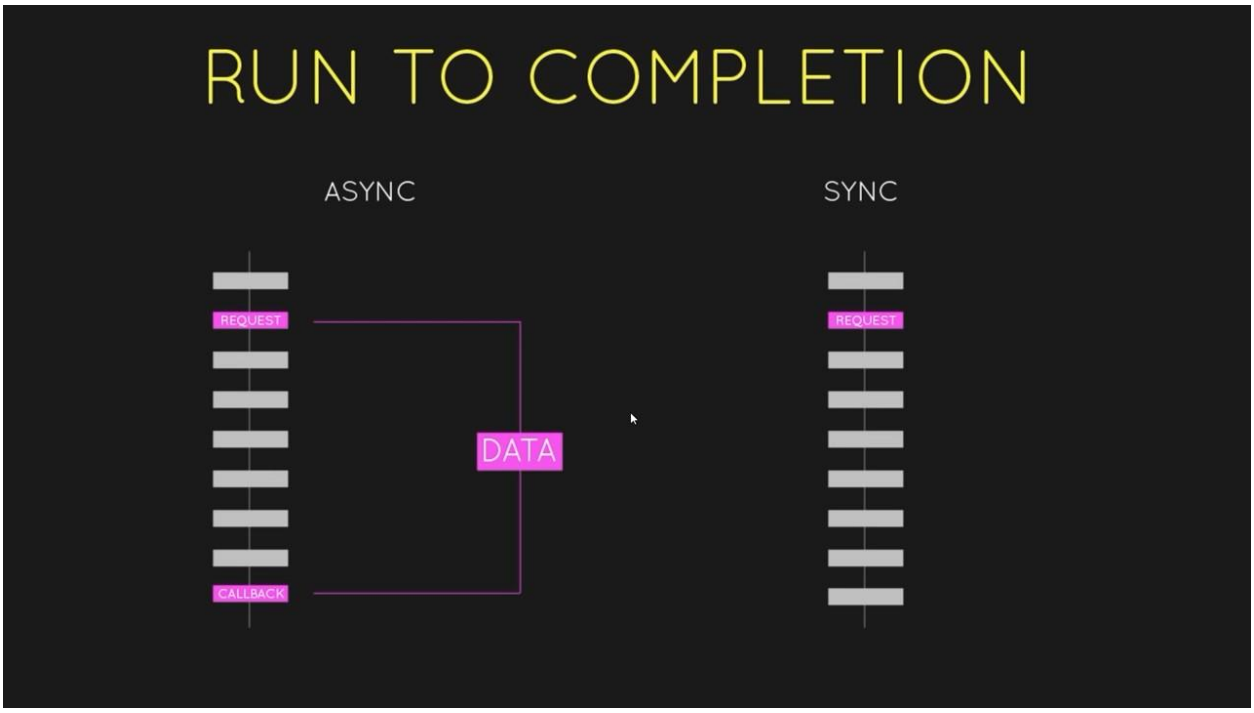
Where its wait until **Call Stack** is empty and then move to it to execute.

Created by: Wael Zoaiter - Github

#4: Event Loop:

The loop that check if the **`Call Stack`** is empty to send the waiting **`callback`** to it.

Asynchronous VS Sunchronous

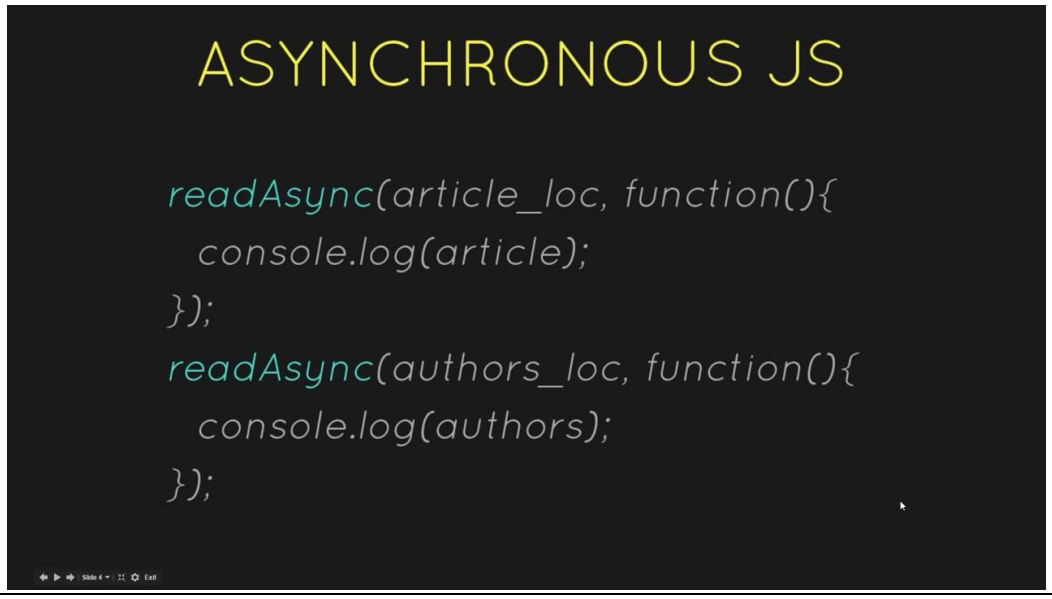


Asynchronous Request

هو أن تنفذ الدالة بحيث لا تعيق تنفيذ الكود don't block the code

ففي حال احتاجت الدالة وقت تقوم بمتابعة الكود وتنفيذ الدالة التالية فتقوم بإنشاء نيسب Thread خارج الجافا سكربت لإحضار البيانات الخاصة بها

وعند اكتمال احضار هذه البيانات يتم وضع دالة استدعاء Callback في نهاية الكود لإكمال الدالة المؤجلة.



Created by: Wael Zoaiter - Github

Synchronous Request

هو أن تنفذ الدوال في الكود بشكل تسلسلي بحيث يتم تنفيذ الدالة التالية بعد انتهاء الدالة التي قبلها بشكل كامل.

Functions

`setTimeout(` () => {callback function}**`,`** time event**`);`**

the time to wait until execute the callback function.

`JSON.stringify(`body**`,`** undefined**`,`** 2**`)`**

To pretty print JSON objects, in depth 2.

`encodeURIComponent(`string**`)`**

to convert string into encoded URI for the browser.

`decodeURIComponent(`encode URI**`)`**

to convert encoded URI into string.

Defenitions

`Callback function:`

it's a function that gets passed as an argument to another function and its executed after some event happens.

e.g. setTimeout(callback, time event)

Created by: Wael Zoaiter - Github

Promises

MANAGE ASYNCHRONOUS COMPUTATIONS

What is it?

Promise is an object that represent an action that hasn't finish yet.

Promise is a placeholder for something that will happen in the future (grab the data and return it to us).

How it works?

As soon as this Asynchronous request is made is return a Promise object straight away before this Data is retrieve and come back to us,

And within this Promise object we can register callback that will run when the request complete.

Functions

```
1  var somePromise = new Promise((resolve, reject) => {
2    setTimeout(() => {
3      resolve('Hey. It worked!');
4      // reject('Unable to fulfill promise');
5    }, 2500);
6  });
7
8  somePromise.then((message) => {
9    console.log('Success: ', message);
10 }, (errorMessage) => {
11   console.log('Error: ', errorMessage);
12 });
```

.then(resolve function**,** reject function**)**

Note: we can only resolve or reject the promise once, and we can't reject a promise that it have been resolved before or the opposite.

Created by: Wael Zoaiter - Github

Asynchronous Add Function

```
var asyncAdd = (a,b) => {
  return new Promise( (resolve, reject) => {
    setTimeout( () => {
      if (typeof a === 'number' && typeof b === 'number') {
        resolve(a + b);
      } else {
        reject('Arguments must be numbers.');
      }
    }, 2000);
  });
};

asyncAdd(5,7).then((result) => {
  console.log('Result: ', result);
}, (errMsg) => {
  console.log(errMsg);
});
```

Error for Chain of asynchronous code

```
asyncAdd(5,7)
  .then((result) => {
    console.log('Result: ', result);
    return asyncAdd(result,20);
  }).then((result) => {
    console.log('The result is:', result);
  }).catch( (errMsg) => {
    console.log(errMsg);
});
```

When we have a chain of asynchronous code we define **`.catch()`** method to catch the error in all of them.

Created by: Wael Zoaiter - Github

Weather app with Promises

```
var encodedAddress = encodeURIComponent(argv.address);
var encodedUrl = `http://maps.googleapis.com/maps/api/geocode/json?address=${encodedAddress}`;

axios.get(encodedUrl).then((response) => {
  if(response.data.status === 'ZERO_RESULTS') {
    throw new Error('Unable to find that address.'); // pass in err.message
  }
  var lat = response.data.results[0].geometry.location.lat;
  var lng = response.data.results[0].geometry.location.lng;
  var weatherUrl = `https://api.darksky.net/forecast/fdd82e47ab90dd24af16f8ad7356a3ca/${lat},${lng}`;
  console.log(response.data.results[0].formatted_address);
  return axios.get(weatherUrl);
}).then((response) => {
  var temperature = ((5/9)*(response.data.currently.temperature-32));
  var apparentTemperature = ((5/9)*(response.data.currently.apparentTemperature-32));
  console.log(`It's currently ${temperature}. It feel like ${apparentTemperature}`);
}).catch((err) => {
  if(err.code === 'ENOTFOUND') {
    console.log('Unable to connect to API servers.');
  } else {
    console.log(err.message);
  }
});
```

Created by: Wael Zoaiter - Github