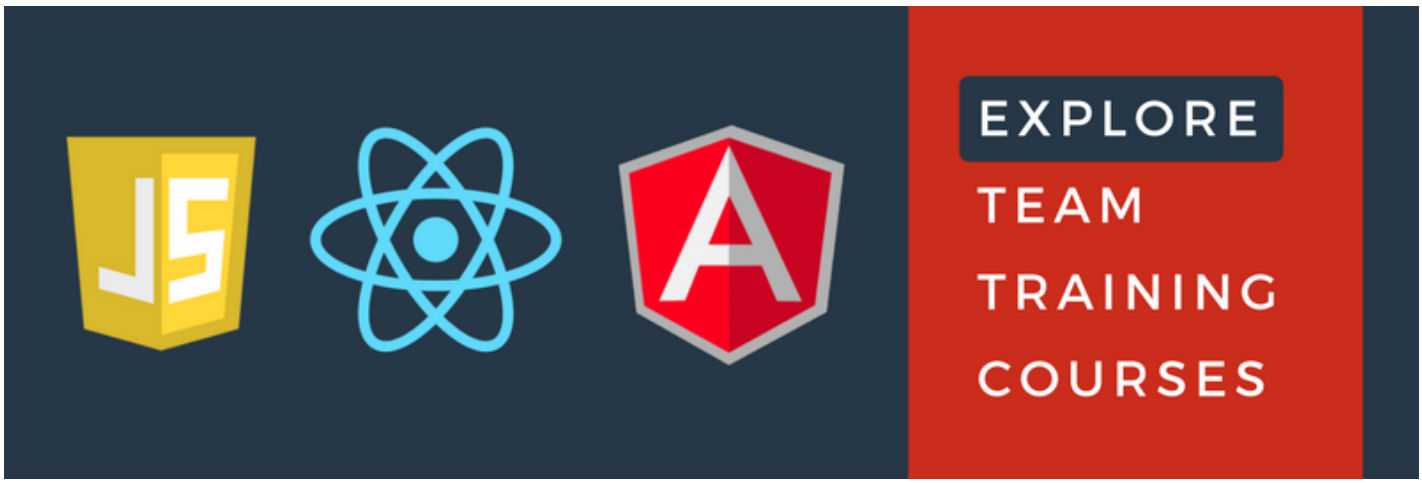
Using JavaScript ASync Functions



Async stands for asynchronous and it is (aptly) the opposite of synchronous programming. To get a better grasp on Async functions, you need to first fully understand synchronous code. Simply put, synchronous code means “many statements in a sequence”. Each statement is executed one after the other. This also means that a statement has to wait before the other one is finished.

```
1 console.log(‘1st’);
2 console.log(‘2nd’);
3 console.log(‘3rd’);
```

The previous code above will execute and output the “1st”, “2nd”, “3rd” strings in a row to the console. Simple. It is written synchronously.

Asynchronous code will execute statements outside of the main flow. This allows for async functions to be executed right away without having to wait. Developers see both states of code. An example of Async programming is shown with a simple jQuery function:

```
console.log('1st');
jQuery.get('page.html', function (data) {
    console.log("2nd");
});
console.log('3rd');
```

utput from the example above will be different. “1st”,”3rd”, “2nd”. This is due to unction passing through jQuery.get and not being immediately called. Instead the function has to wait for jQuery to fetch the page before executing.

Async Code: Benefits & Uses

Once the code had run, there is a chance that synchronous code can block future executions. This happens because it is waiting for prior statements to finish. Basically, this means that the UI can become unresponsive until a function has finished. This should be avoided at all costs as it can damper the user experience.

A developer named Brian Terlson, is working on implementing native Async functions into ECMAScript. Currently, it’s in proposal mode and in step three(candidate) of being implemented part of ES7.

Variants of async functions currently exist. The unique keyword is async before every declaration. Here is the following syntax:

- Async function declarations: async function foo() {}
- Async function expressions: const foo = async function () {};
- Async method definitions: let obj = { async foo() {} }
- Async arrow functions: const foo = async () => {};

Generally, you’ll want to use asynchronous code to perform expensive, time consuming operations. It wouldn’t be used for something simple like changing the CSS class of an element. Here is an example of another async function, but this time with the keyword async in front of it.

```
async function asyncFunc() {
    return xyz;
```

```
3 }
4
5 asyncFunc()
6 .then(x => console.log(x));
7     // xyz
```

e the ".then" operator. An async function is passed through the argument by ething called a Promise implementation. According to its definition, a promise represents "a value which may be available now, or in the future, or never."

A promise is in three different states:

- pending: starting state, not yet fulfilled or rejected.
- fulfilled: the operation has finished successfully.
- rejected: the operation has failed.

So to recap, asynchronous code should be used for when synchronous code won't cut it. It adds a new degree of complexity that will keep your user experience smooth and flowing. Your application or website stays responsive, and reduces idle time for the end user. Native implementation of Async functions will make this a ubiquitous way to execute aysnc functions.

Mike Colagrossi

December 3, 2016

by Mike Colagrossi in

ES5, es6, JavaScript

Comments

Comment

Name

Email

Add Comment