

Async Programming Constructs in ES2017

What's the hype?

Ying Ka Ho
@yingkh_tweets

async/await in ES2017

What's the hype?

Ying Ka Ho
@yingkh_tweets

Async function declaration:

```
async[no LineTerminator here]functionBindingIdentifier[?Yield, ?Await](FormalParameters[~Yield, ?Await]){  
    AsyncFunctionBody  
}
```

```
[+Default]async[no LineTerminator here]function(FormalParameters[~Yield, ?Await]){AsyncFunctionBody}
```

Await expression:

```
await UnaryExpression[?Yield, +Await]
```

AJAX

- Event handlers

Messy!

- Callbacks

Readable?

Promise



Promises and Futures

- Callbacks [`promise.then(function)`]
- Generators [`yield`]



“Only old-school async requires you to write your code ‘inside out’.”

- Stephen Cleary, who wrote his first async program in C++ in year 2001.

<https://blog.stephencleary.com/2012/09/an-async-horror-story.html>

Typical async function in js

```
async function callFunctionDelayed(){
    var response = await getRandomQuotesDelayed();
    document.getElementById('content').innerHTML = response;
}
```

Typical async function in Typescript

```
async function printDelayed(elements: string[]) {
    for (const element of elements) {
        await delay(200);
        console.log(element);
    }
}

function delay(milliseconds: number) {
    return new Promise<void>(resolve => {
        setTimeout(resolve, milliseconds);
    });
}
```

The screenshot shows a web browser displaying a HackerNoon article. The browser's address bar shows the URL: <https://hackernoon.com/6-reasons-why-javascripts-async-await-blows-promises-away-tutorial-c7ec10518dd9>. The HackerNoon logo is in the top left, and navigation links (HOME, JAVASCRIPT, VC, API, CAT, MVP, ARCHIVE, ABOUT, SAVE NET NEUTRALITY (PETITION)) are in the top right. The article is by Mostafa Gaafar, a Software Engineer, with a 'Follow' button. The title is '6 Reasons Why JavaScript's Async/Await Blows Promises Away (Tutorial)'. The text explains that Node.js now supports async/await out of the box since version 7.6 and provides examples of why to adopt it. It includes an update about Node 8 LTS and an edit about embedded code. A section titled 'Async/await 101' provides a quick intro for newcomers. At the bottom, there is a 'GET UPDATES' button and a note about signing up for Medium.

6 Reasons Why JavaScript's Async/Await Blows Promises Away (Tutorial)

In case you missed it, Node now supports async/await out of the box since version 7.6. If you haven't tried it yet, here are a bunch of reasons with examples why you should adopt it immediately and never look back.

[UPDATE]: [Node 8 LTS](#) is out now with full Async/Await support.

[EDIT]: It seems that the embedded code on gist does not work on medium native app, but it works on mobile browsers. If you are reading this on the app, tap on the share icon and choose "open in browser" in order to see code snippets.

Async/await 101

For those who have never heard of this topic before, here's a quick intro

- Async/await is a new way to write asynchronous code. Previous options for asynchronous code are callbacks and promises.
- Async/await is actually built on top of promises. It cannot be used with

Never miss a story from **Hacker Noon**, when you sign up for Medium. [Learn more](#)

GET UPDATES

- Error handling
- Conditionals
- Intermediate values
- Error stacks
- Debugging

<https://hackernoon.com/6-reasons-why-javascripts-async-await-blows-promises-away-tutorial-c7ec10518dd9>

HOGWARTS SCHOOLS OF WITCHCRAFT AND WIZARDRY

....

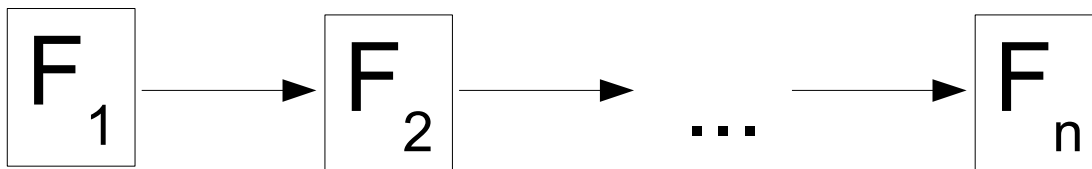
Terms begins on 1 September. We [await](#) your owl by no later than 31 July

Yours Sincerely,

Minerva McGonagall
Deputy Headmistress

Javascript

```
async function callFunctionDelayed(){  
    var response = await getDataFromServerDelayed();  
    var result = computeForResult(response);  
    return result;  
}
```



C#

```
public async Task<string> CallFunctionAsync(){  
    var response = await GetDataFromServerAsync();  
    var result = ComputeForResult(response);  
    return result;  
}
```

Bad Examples

```
async function getRandomQuotesDelayed(){
  var request = new XMLHttpRequest();
  return await new Promise(function(resolve, reject){
    request.onreadystatechange = function() {
      if (request.readyState === XMLHttpRequest.DONE){
        if (request.status === 200){
          resolve(request.responseText);
        }
        else{
          reject(new Error('There is a problem in requesting for a webpage.'));
        }
      }
    }
    request.open("GET", 'http://ron-swanson-quotes.herokuapp.com/v2/quotes');
    request.send();
  });
}
```

Bad Examples

```
async function computeDelayed(){  
    var result = synchronousComputations();  
  
    return await new Promise(function(resolve, reject){  
        resolve(result);  
    });  
}
```

```
async function callAsyncFunctionDelayed(){  
    synchronousWork();  
    var result = await asyncFuncDelayed();  
    return result;  
}
```

Notes on async/await

- The understanding of the meaning of 'await' to prevent the misuse of the keyword.
- Using 'async' keyword when you need to return the result in an asynchronous function.

Notes on learning async programming

- Don't just stop on async/await.
- There's reactive programming, actor programming, parallel programming and dataflow programming
- The plot thickens when more than one threads are involved.
- Don't forget about how useful is async on IO.

Summary

- Async programming constructs before ES2017
- `async/await` constructs
- Notes on async programming.