## **ES?**

### **Introduction**

We’ve been throwing around the term ES6 since our very first lessons, but we haven’t taken the time to properly explain what it means or to investigate the implications of it in our code.

Put simply ES6 is a version of JavaScript that was officially released in the summer of 2015. It included *many* new features that make writing JavaScript much easier and cleaner. If you have been following our lessons you have already been learning these new features because, well, ES6 is *just JavaScript*.

You have probably also come across articles talking about features in ES7 or ES8 or ES2015 or ES2017 etc. Part of the confusion here is that right after the release of ES6, the committee that makes these decisions changed the naming scheme from ‘version numbers’ (ES5, ES6, ES7 etc.) to ‘release years’ (ES2015, ES2016, ES2017 etc.)

* [This article](https://codeburst.io/javascript-wtf-is-es6-es8-es-2017-ecmascript-dca859e4821c) provides a nice clean explanation and timeline of the various ECMAScript releases.
* [This document](https://github.com/lukehoban/es6features) outlines all the new features that showed up in ES6. As we’ve mentioned you’ve already been using many of these, though there are a few we haven’t specifically covered yet.

The *problem* with JavaScript constantly updating and adding features is that it sometimes takes web-browsers a while to catch up and implement new features once they’ve been released. At the current time all modern browsers (Chrome, Firefox, Safari and Edge) support *all* of ES6, and *most* of ES7, but older browsers (various versions of Internet Explorer for instance) do not. This means, unfortunately, that if you write code that uses these new features it will not run in browsers that do not support it.

For most of us, this has not been an issue because you are almost definitely using a new browser that automatically updates itself when a new version is released. But in the real world, if you’re selling products to customers you can’t control which browsers people will use to connect to your site.

Fortunately there *is a solution* to this problem. [Babel](http://babeljs.io/) is a tool that takes your modern JavaScript code and translates it to code that older browsers can understand. It can be used from the command line with a simple command, and can also easily be added to your webpack configuration. With the babel-loader.

In all honesty, this is not something that you are going to *need* to worry about on every project you’re starting. All the ES6 features are present in the large majority of browsers used worldwide. But JavaScript is constantly changing, and as new features are announced and released, you can use Babel to try them out, often before they’re available in *any* browser!

* Follow the instructions [here](https://github.com/babel/babel-loader) to install the babel-loader and use it with webpack. If you’ve already got webpack up and running in a project, adding babel is a cinch!