# **AtScript has been replaced with TypeScript; see the** [**blog post**](http://blogs.msdn.com/b/typescript/archive/2015/03/05/angular-2-0-built-on-typescript.aspx)**.**

# 

# 

# Historical reference: AtScript (was "ES6 +A") Q&A

We received lots of great questions about using ECMAScript6 (ES6) as the core language for AngularJS 2.0 and about our unofficial annotation extension on top of ES6.

## tl;dr

AngularJS 2.0 itself will be written with AtScript, but then transpiled to ES5. You can write your AngularJS 2.0 app with ES5, ES6, AtScript, CoffeeScript, TypeScript, etc. Angular 2.0 apps will run in today's browsers, and even better as ES6 features land.

## What is ES6?

ES6 is 6th version of the ECMAScript standard specification for what is more commonly known as just ‘JavaScript’. Here’s [the current draft of the ES6 spec](https://people.mozilla.org/~jorendorff/es6-draft.html) if you’d like to read more. Browsers you know and love today implement ES5 or [ES5.1](http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf).

Many of ES6's features are available in browsers today. There’s a nice [compatibility chart](http://kangax.github.io/es5-compat-table/es6/) that you can check out to track progress. Until all the features are available, we’ll use the [Traceur](https://github.com/google/traceur-compiler) transpiler to turn Angular’s ES6 implementation into ES5 that runs in today’s browsers.

## What is "+A" aka [AtScript](https://docs.google.com/a/google.com/document/d/11YUzC-1d0V1-Q3V0fQ7KSit97HnZoKVygDxpWzEYW0U/edit#)?

"+A" means "plus annotations." Annotations are an optional feature on top of ES6. We want "+A" to stand alone from "ES6" because annotations are not an official part of the ES6 specification. As standards evolve, we will adapt accordingly.

## Q: Can I use AngularJS 2.0 without using the transpiler?

Yes! You don't have to use ES6 to write your AngularJS 2.0 app. We hope to make it so awesome that you’ll want to. But if not, here’s how you’d do it.

Given this example from AngularJS today:

### ES5 (Today's JavaScript) and AngularJS v1.x

|  |
| --- |
| var module = angular.module('myApp');  module.directive('myDirective', function() {  return {  restrict: 'A',  link: function(scope, element, attr) {  ...;  }  };  }); |

Below are code samples in languages that compare how same thing can be expressed with and without using the new types and annotations.

### ES6 +A / AtScript

|  |
| --- |
| import {DecoratorDirective, Inject} from 'angular';  **@DecoratorDirective({**  **selector: '[my-directive]'**  **})**  export class MyDirective {  **@Inject()**  constructor(element:**Element**) {  ...  }  } |

### 

### ES6

|  |
| --- |
| import {DecoratorDirective, Inject} from 'angular';  export class MyDirective {  constructor(element) {  ...  }  }  MyDirective.annotations = [  **new DecoratorDirective({ selector: '[my-directive]' }),**  **new Inject(Element)**  ]; |

### ES5 (Today's JavaScript) + RequireJS

|  |
| --- |
| require(['angular']  function(ng) {  var MyDirective = function(element) {  ...  }  MyDirective.annotations = [  **new ng.DecoratorDirective({ selector: '[my-directive]' }),**  **new ng.Inject(Element)**  ];  }); |

### TypeScript

|  |
| --- |
| import ng = require('angular');  export class MyDirective {  public static annotations = [  **new ng.DecoratorDirective({ selector: '[my-directive]' }),**  **new ng.Inject(Element)**  ];  constructor(element) {  ...  }  } |

### CoffeeScript + RequireJS

|  |
| --- |
| require ['angular'], (ng) ->  class MyDirective  @annotations = [  **new ng.DecoratorDirective selector: '[my-directive]',**  **new ng.Inject Element**  ]  constructor: (@element) ->  ... |

## Q: Why will AngularJS use ES6?

Angular's goal is to give you tools to best express your application. ES6 has several new language-level constructs that we want to make easy to use with Angular. Examples include...

**Class syntax:** Though you can write classes in a roundabout fashion in ES5 today, ES6 has a formal way to make a class. All the stuff you had to do manually before gets done with syntax that expresses it. Here’s a [nice example](http://dstrunk.com/ecmascript-6-features-classes/). Scroll to the bottom for Angular-specific samples.

**Modules:** We like [modules](http://www.infoq.com/news/2013/08/es6-modules) as a first class construct in the browser. We built our own module system in Angular 1.x only because at the time there wasn’t a single one that would work for everyone. Eventually ES6 modules will ship with every browser, we can just use the thing that standards are working towards.

## Q: Why aren’t you waiting for ES6 to be available in browsers?

We think ES6 is a significant improvement over ES5 and developers will benefit from ES6 sooner if it is more accessible. We want to help ES6 adoption. While building Angular in ES6, we also want to create tools and practices that let others get in the game sooner.

We’d also like to solve a timing problem with adoption. Frameworks like AngularJS are the base on which other code gets written. We're building AngularJS in ES6 now so you’ll have a framework that’s ready to take full advantage of ES6 as browsers add support.

We’re building AngularJS 2.0 from scratch. Changes in ES6 are so significant that it doesn’t make sense to build Angular on ES5 today and then change to ES6 in a year.

In the mean time, we will continue to improve and support AngularJS 1.x.

## Q: Why extend ES6 with type and data annotations (+A)?

At a high level, "annotation" just means "a way of specifying metadata about a related object."

AngularJS always had annotations. The directive definition object and the array annotations for DI are two examples. These are annotations are implemented with existing parts of the ES5 language.

We think annotations are so ubiquitous that they deserve their own syntax. With +A annotations, we can say what we mean in a syntax designed for metadata.

Annotations aren’t yet in the current ECMAScript spec, but we’re in good company. Types and annotations were in the ES4 specification, ActionScript, Java, and even JavaScript (within comments) via Google’s Closure library.

## Q: Why aren’t you waiting for a standardized annotations feature?

In the spirit of prollyfills, the [Extensible Web Manifesto](http://extensiblewebmanifesto.org/), and [How The Web Should Work](http://smus.com/how-the-web-should-work/), we’d like to show how useful this feature can be in JavaScript to make it easy for the standards bodies to see syntax that works in real apps.

As the ECMAScript standard evolves, we’ll evolve along with it.

## Q: What does this mean for my editor, linter, and other tools?

We’re working with other teams on support for lint/hint tools and with editor vendors on syntax and refactoring support. We’ll keep you posted on progress.

## Got more questions?

Please keep them coming. You can comment in this doc or find us Angular team members on Google+ or Twitter ([Brad Green](https://twitter.com/bradlygreen), [Misko Hevery](https://twitter.com/mhevery), [Brian Ford](https://twitter.com/briantford), [Igor Minar](https://twitter.com/igorminar), [Vojta Jina](https://twitter.com/vojtajina), etc.).