

TypeScript

future and past

Notice

Codes in this slide might be invalid

Notice

**Codes in this slide might be invalid
Even in the future**

Type

- JavaScript is dynamic type
- No type check at compile time and run time

```
var hey
```

```
hey = 1
```

```
hey = 'this is string'
```

```
hey = false
```

```
int hey
```

```
hey = 1
```

```
hey = 'this is string' // Error
```

```
hey = false // Error
```

Pros

- **Compiler optimization**
- **Reliability for large scale app**
- **IDE support**

過去

ECMAScript 4

- Lots of new features
- Type annotation
- Static type check

```
var hey:number
```

```
hey = 1
```

```
hey = 'this is string' // TypeError
```

```
hey = false // TypeError
```

```
var ho = {  
    id: 123,  
    desc: "hoho"  
} : {  
    id: int,  
    desc: string  
}
```

```
type Tuple = {  
  id:    int,  
  desc:  string  
}
```

```
var ho = {  
  id:    123,  
  desc:  "hoho"  
} : Tuple
```

ECMAScript 4

- Deprecating to be ECMAScript standard
- Live in ActionScript 3
 - Flash, Flex

現在

- **Type in compile to JavaScript languages**

TypeScript

learn play download interact

TypeScript lets you write JavaScript the way you really want to.
TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
Any browser. Any host. Any OS. Open Source.

Get TypeScript Now →

Scalable

TypeScript offers classes, modules, and interfaces to help you build robust components.

These features are available at development time for high-confidence application development, but are compiled into simple JavaScript.

TypeScript types let you define interfaces between software components and to gain insight into the behavior of existing JavaScript libraries.

TypeScript

- Microsoft, 2012
- Add type and several features to JavaScript(ES5)
- JavaScript Superset

TypeScript

Type

Class

Generics

Module

Type

- Optional type annotation
- Compile time type check
- Type definition file

```
var hey:number
```

```
hey = 1
```

```
hey = 'this is string' // Error
```

```
hey = false // Error
```

```
var hey:number
```

```
hey = 1
```

```
hey = 'this is string' // Compile Error
```

```
hey = false // Compile Error
```

```
var hey
```

```
hey = 1
```

```
interface Tuple {  
    id:    number;  
    desc: string;  
}
```

```
var ho:Tuple = {  
    id:    123,  
    desc: "hoho"  
}
```

Definition File

- Like C++ header file
- Define library interface
- File extension: `.d.ts`
- Work with Visual Studio, TernJS



DefinitelyTyped

The repository for high quality TypeScript type definitions

Usage

Include a line like this:

```
/// <reference path="jquery/jquery.d.ts" />
```

Get the definitions

- [GitHub repository](#)
- [NuGet package manager](#)
- [TypeScript Definition manager](#)

Contributing

See the [contribution guide](#)

News

- Add a [badge](#) to your library
- TypeScript [directory](#) restructured

700+ libs

TSD TypeScript Definition manager for DefinitelyTyped

Fork me on GitHub

- Get TSD
- Contribute
- DefinitelyTyped
- NuGet
- Readme
- Issues

Install TSD globally using [npm](#):

```
$ npm install tsd -g
```

Check the [README.md](#) for usage info.

TSD is a package manager to search and install [TypeScript](#) definition files directly from the community driven [DefinitelyTyped](#) repository.

Got design skills? TSD is looking for a creative to improve this site. Leave a message [here](#) if you like Grunt, Github and Vue.

screenshots

- [help](#)
- [async](#)
- [angular](#)

browse

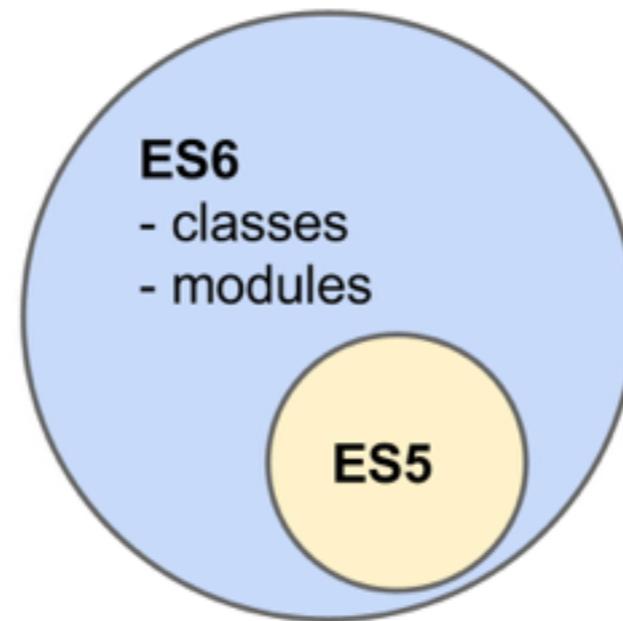
Project	Name	Version	View	Homepage
Finch	Finch	0.5.13	github	project

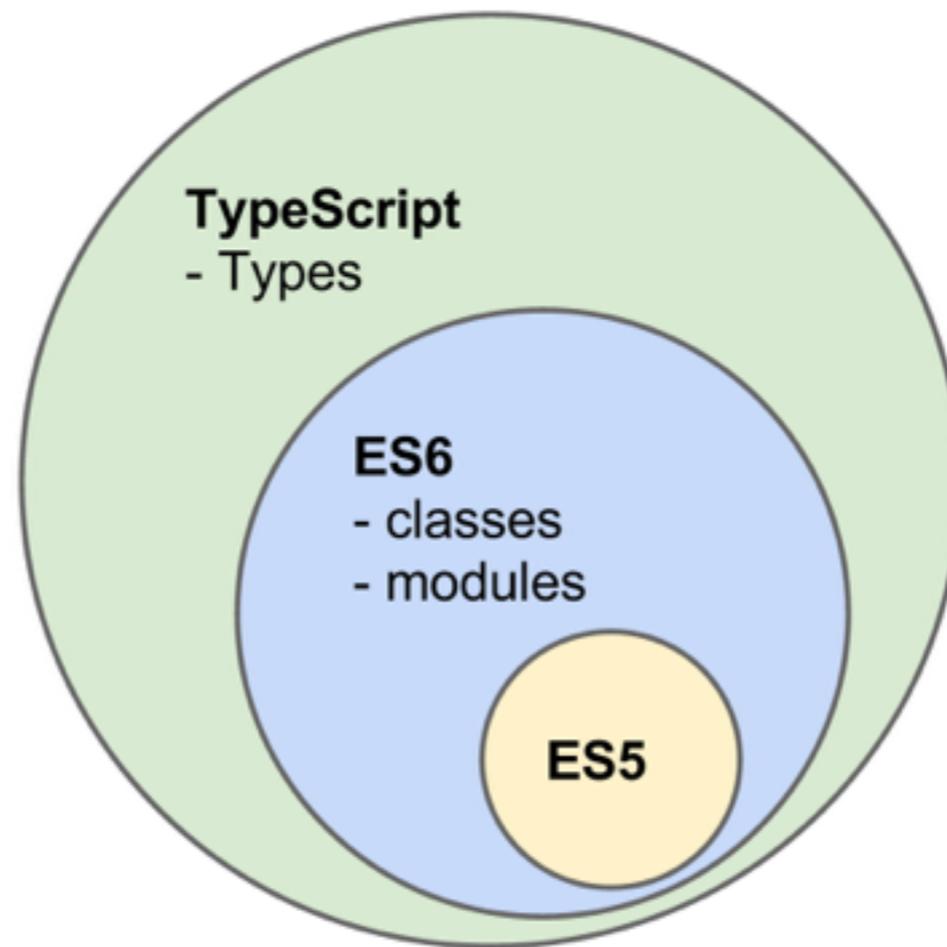
Projects

- **AngularJS 2**
- **Asana**
- **Immutable.js**
- **Shumway by Mozilla**

TypeScript 1.5+

- **Align to ECMAScript 6**
 - **Use native module and class**
- **More ECMAScript 6 features**





**Angular Team
not Satisfy**

AtScript

- **Google Angular Team, 2014**
- **Annotation, Introspection**
- **At means @**

AtScript Primer

Status: Draft

Author: misko@google.com

This document is published to the web as part of the public [Angular Design Docs](#) folder

Goal

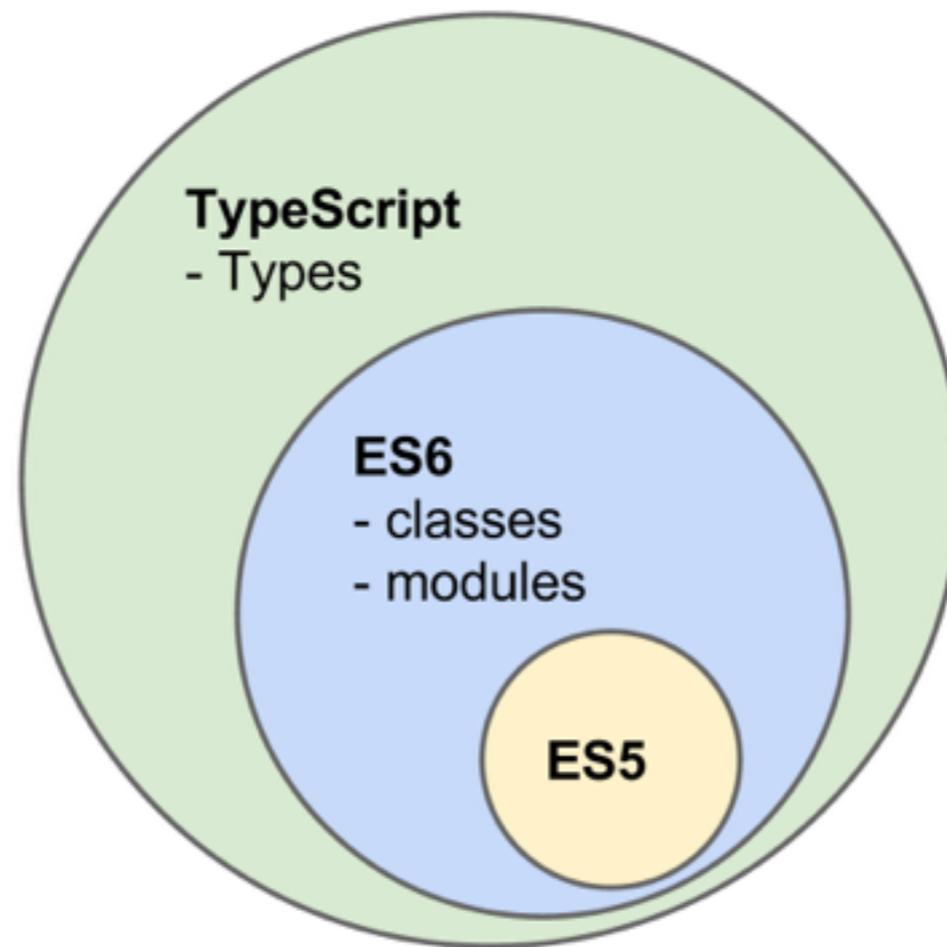
JavaScript, the de facto language of the browser, has a large, thriving community. However, it is missing some features which would make it much more manageable for large-scale application development. The goal of AtScript is to enhance the language with these missing features without infringing upon its current capabilities.

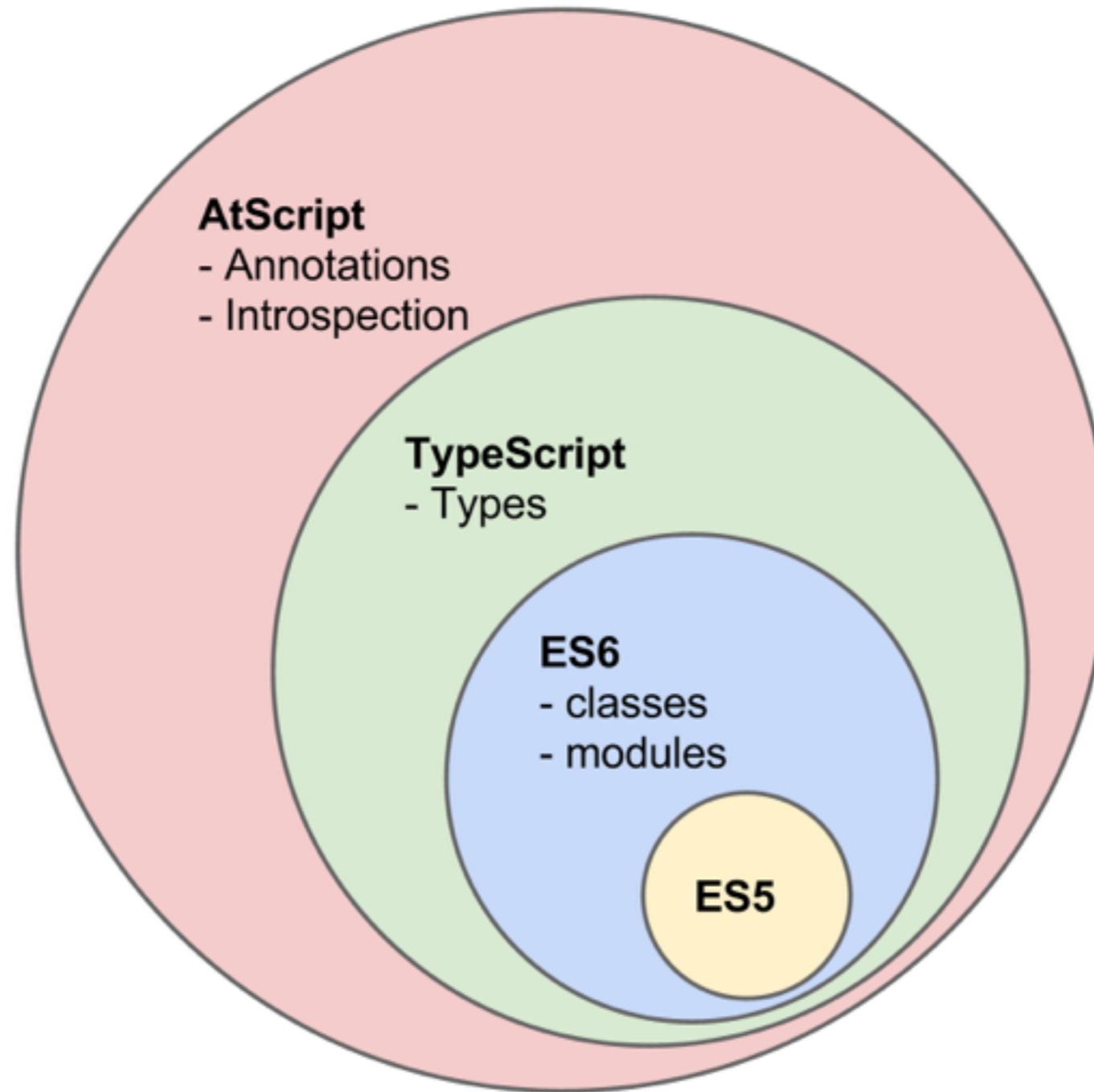
Enhancements

- **Type Annotations:** Types allow us to explicitly express contracts between different parts of the system and to give these contracts names. For large teams with large codebases, where no single person knows the whole codebase, types become a key way to discuss and reason about the collaboration between different components of the system.
- **Field Annotations:** Types are not only about contracts (API) but also about structure. In JavaScript the fields are implicit; they are created as a side effect of assignment. By giving the developer syntax to explicitly declare the fields, and warning upon implicit creation, we allow them to think about types in a more concrete way. By doing this we in no way sacrifice the flexibility of adding fields dynamically. We simply provide an optional tool to developers who can benefit from being more explicit.
- **Metadata Annotations:** Many things in programming are best expressed in a declarative fashion. Declarative annotations allow frameworks to understand the meaning of your code without forcing a complex inheritance or lifecycle API. Examples where this technique has been used in the past include dependency injection libraries and web frameworks.
- **[Type Introspection](#) with Annotation Support:** When we have annotations, it's important to provide a consistent API that developers and frameworks can leverage to gain access to this information at runtime. The reflection API should also be backwards compatible with ES5 and ES6.

Philosophy

- **ES6 Baseline:** ES6, or ECMAScript 6, is the latest version of the Ecma International language popularly known as JavaScript. It includes many improvements such as a formal class declaration syntax, promises, modules and consistent variable scoping.
- **Backwards Compatibility:** Types, fields, metadata annotations and reflective access need to be added in a way which does not break the existing syntax or semantics of ES6/ES5. AtScript needs to be a more succinct way of expressing what is already possible in these languages. ES6/ES5 code must be valid AtScript code without any changes to the semantics. Any developer who has written ES6/ES5 code can immediately get started with AtScript. In other words, ES6/ES5 are strict subsets of AtScript. All the code you are used to writing today will work without alteration with AtScript, but now you can take advantage of the enhancements to the language which will allow you to express your ideas in a more concrete way.





Annotation

- 標註
- Store meta data
- Accessible in runtime
- Like Java annotation

@Memorize

```
function fib(n) {  
    if (n < 2) { return n }  
    return fib(n - 1) + fib(n - 2)  
}
```

```
function fib(n) {  
  if (n < 2) { return n }  
  return fib(n - 1) + fib(n - 2)  
}
```

```
fib.annotations = [  
  new Memorize()  
]
```

Runtime Readable

- Use ``new`` to create a new instance
- Store under ``annotations``

Introspection

- 内省
- Runtime type check

Runtime Type Check

- No magic
- Add code to check type
- Use `assert.js`

ASSERT.JS



A run-time type assertion library for JavaScript. Designed to be used with [Traceur](#).

- [Basic Type Check](#)
- [Custom Check](#)
- [Primitive Values](#)
- [Describing more complex types](#)
 - [assert.arrayOf](#)
 - [assert.structure](#)
- [Integrating with Traceur](#)

```
import {assert} from 'assert';
```

BASIC TYPE CHECK

By default `instanceof` is used to check the type

```
function fib(n:number):number {  
    if (n < 2) { return n }  
    return fib(n - 1) + fib(n - 2)  
}
```

```
function fib(n) {
  assert.argumentTypes(n, number)
  if (n < 2) {
    return assert.returnType((n), number)
  }
  return assert.returnType(
    (fib(n - 1) + fib(n - 2)), number
  )
}
```

```
function fib(n) {  
    assert.argumentTypes(n, number)  
    if (n < 2) {  
        return assert.returnType((n), number)  
    }  
    return assert.returnType(  
        (fib(n - 1) + fib(n - 2)), number  
    )  
}
```

```
function fib(n) {  
  assert.argumentTypes(n, number)  
  if (n < 2) {  
    return assert.returnType((n), number)  
  }  
  return assert.returnType(  
    (fib(n - 1) + fib(n - 2)), number  
  )  
}
```

```
function fib(n) {  
    assert.argumentTypes(n, number);  
    if (n < 2) {  
        return assert.returnType((n), number);  
    }  
    return assert.returnType(  
        (fib(n - 1) + fib(n - 2)), number  
    );  
}
```

Performance Impact

- Yes, of course
- Only run type check at development time
- Compile to no type check version for production

AtScript Compiler

- Use traceur with options

AtScript Playground

- Traceur environment ready for play

```
{  
  "traceur": {  
    "modules": "amd",  
    "script": false,  
    "types": true,  
    "typeAssertions": true,  
    "typeAssertionModule": "assert",  
    "annotations": true,  
    "sourceMaps": "file"  
  }  
}
```

```
{  
  "traceur": {  
    "modules": "amd",  
    "script": false,  
    "types": true,  
    "typeAssertions": true,  
    "typeAssertionModule": "assert",  
    "annotations": true,  
    "sourceMaps": "file"  
  }  
}
```

**Facebook want
Their Own Solution**

Flow

- Facebook's static type checker
- Compatible with TypeScript's syntax
- Several difference

Flow | Flow | A static type checker

flowtype.org

flow

DOCS ABOUT SUPPORT BLOG GITHUB



flow

A STATIC TYPE CHECKER FOR JAVASCRIPT

[GET STARTED](#)

What is Flow?

Flow is a static type checker, designed to find type errors in JavaScript programs:

```
1 /* @flow */
2 function foo(x) {
3   return x * 10;
4 }
5 foo('Hello, world!');
```

```
$> flow
hello.js:5:5,19: string
This type is incompatible with
hello.js:3:10,15: number
```

Flow also lets you gradually evolve JavaScript code into typed code:

Difference

- Doesn't compile ES6 to ES5
- Scalability, flow analysis
- More types, ex: maybe, non-nullable
- Integrated with JSX



Avik Chaudhuri
Software Engineer at Facebook

12:51 / 46:25

JavaScript Testing and Static Type Systems at Scale - @Scale 2014 - Web

@Scale
訂閱 2,408

9,630

即將播放 自動播放

Keynote - @Scale 2014
由@Scale建立
觀看次數：853
20:42

Open Lecture by James Bach on Software

未來

Old Proposals

Types

Old proposal (2009)

Guards

Convenient syntax for Trademarks

Trademarks

Newer proposal (2011) by Waldemar Horwat

Old Proposals

Types

<http://wiki.ecmascript.org/doku.php?id=strawman:types>

Guards

<http://wiki.ecmascript.org/doku.php?id=strawman:guards>

Trademarks

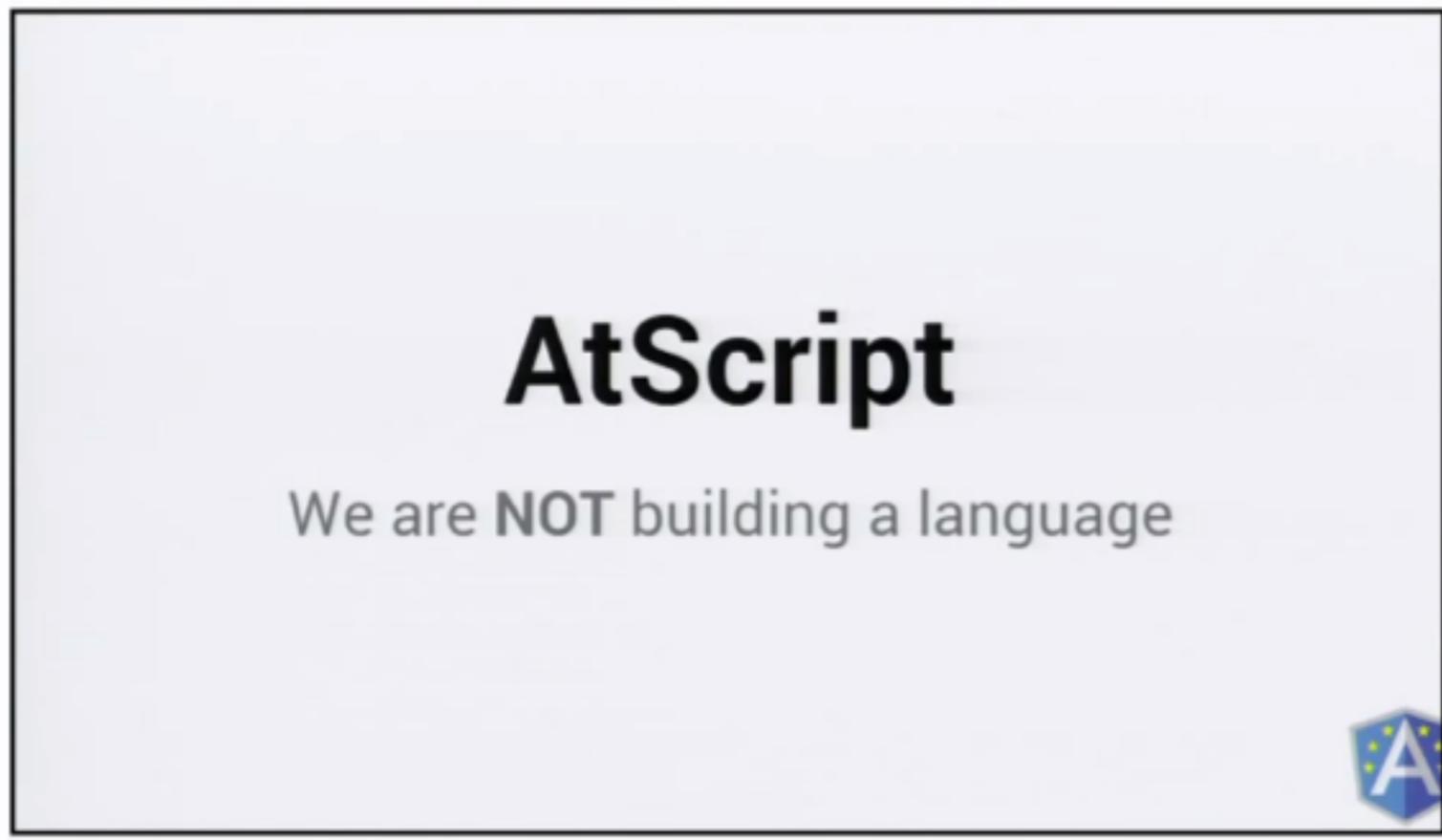
<http://wiki.ecmascript.org/doku.php?id=strawman:trademarks>

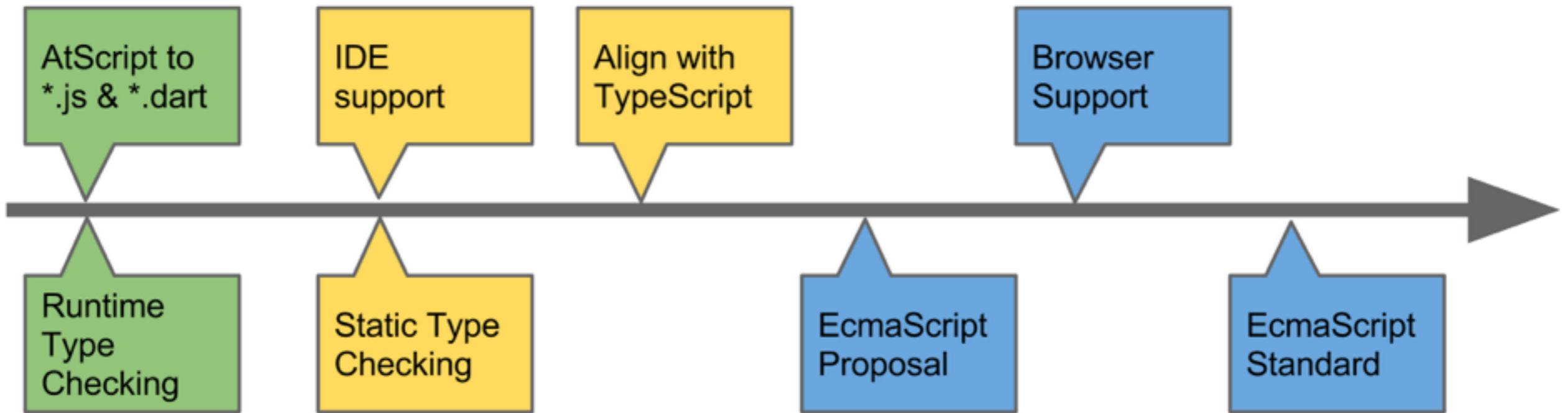
Type

```
var ho = {  
  id: 123,  
  desc: "hoho"  
} : {  
  id: int,  
  desc: string  
}
```

Guard

```
var ho = {  
  id :: Integer : 123,  
  desc :: String : "hoho"  
}
```





Now

- **AtScript no more activity**
- **Angular 2.0 uses TypeScript**
- **TypeScript might merge to ES.next**

- **Microsoft, Google, Facebook are talking together about type in ECMAScript**

SoundScript

- **V8 experiment**
- **TypeScript compatible syntax**
- **`--strong-mode`**

```
"use stricter+types";
```

Experimental New Directions for JavaScript

Andreas Rossberg, V8/Google

Thursday, January 29, 15

One more thing

Annotation

- **Metadata will be parse and use by compiler and runtime**
- **Type annotation tells the variable data type to compiler**

- How about we want declare some characteristic on objects, methods?
 - memorize
 - readonly
 -

Decorator

- Syntax sugar
- Looks like annotation
- Like Python decorator
- by Yehuda Katz

Decorators for ES7

(Yehuda Katz)

Slides

YK: Presenting aspects of common use cases not yet covered by ES6 `class`.

Knockout.js example (compute the value of a property)

WH: Do you want to use functors to produce functions that are per-class (i.e. on the prototype) or per-instance?

AWB: Per instance wants to be handled in the constructor

YUI example (a readonly property)

LH/YK: Sometimes you want to say a method is readOnly

AWB: No declarative way to describe the per instance state

Angular example

LH: (explanation) when I declare a class, I also want to register it with some other system

ES6 Experiments: Angular

```
@NgDirective('[ng-bind]')  
class NgBind {
```

<https://github.com/jonathandturner/brainstorming>

We should probably revise and post it to a more official place.



wycats commented 16 days ago

I'm planning on extracting the salient parts into a concrete TC39 proposal this weekend and present it at the next TC39 in a few weeks.

This was referenced 16 days ago

Decorators #2249

[Open](#)

Remove AtScript support in favor of Typescript 1.5 aurelia/metadata#8

[Open](#)

Proposal for porting React's Mixin APIs to a generic primitive facebook/react#1380

[Open](#)

[Sign up for free](#)

to join this conversation on GitHub. Already have an account? [Sign in to comment](#)



Browser window showing the GitHub repository page for `wycats/javascript-decorators`. The address bar displays `https://github.com/wycats/javascript-decorators`.

The repository page includes the following elements:

- Repository Name:** `wycats / javascript-decorators`
- Statistics:** 5 commits, 1 branch, 0 releases, 2 contributors.
- Branch:** `branch: master`
- Commit History:**

File	Commit Message	Time Ago
<code>INITIALIZER_INTEROP.md</code>	Add explainer on interop with declarative props	14 days ago
<code>README.md</code>	Update README.md	15 days ago
- Code Section:** Includes links for `Code`, `Issues` (2), `Pull requests` (2), `Pulse`, and `Graphs`.
- Clone Options:** `HTTPS clone URL` (`https://github.com`), `Clone in Desktop`, and `Download ZIP`.

The `README.md` content is partially visible:

Summary

Decorators make it possible to annotate and modify classes and properties at design time.

While ES5 object literals support arbitrary expressions in the value position, ES6 classes only support literal function expressions. Decorators restore the ability to run code at design time while maintaining

<https://github.com/wycats/javascript-decorators>

```
class M {  
    @memorize  
    fib(n) {  
        if (n < 2) { return n }  
        return this.fib(n - 1)  
            + this.fib(n - 2)  
    }  
}
```

```
class M {  
    @memorize  
    fib(n) {  
        if (n < 2) { return n }  
        return this.fib(n - 1)  
            + this.fib(n - 2)  
    }  
}
```

```
var M = (function () {
  class M {
    fib(n) {
      if (n < 2) { return n }
      return this.fib(n - 1) + this.fib(n - 2)
    }
  }
}

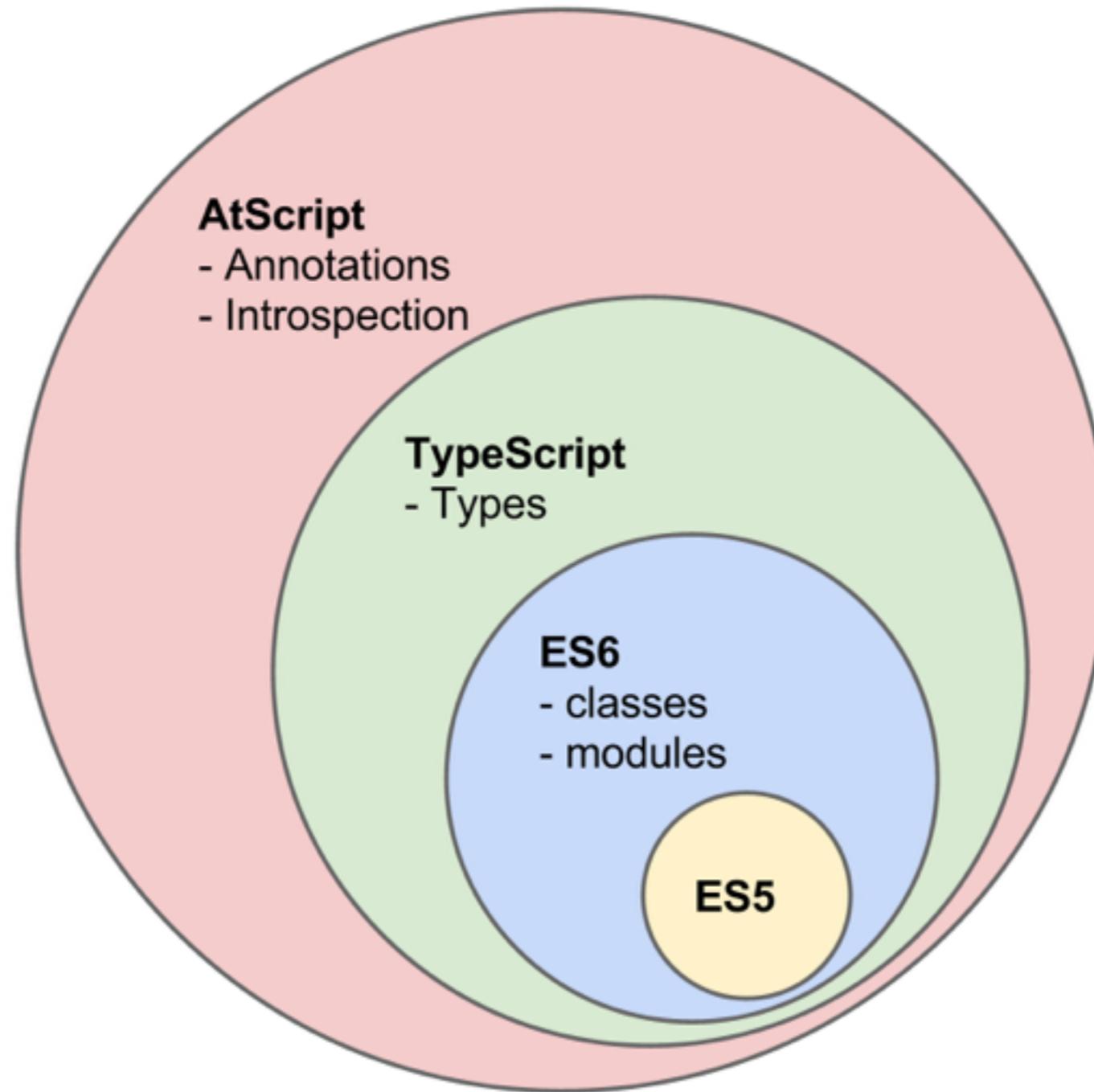
var _temp
_temp = memorize(Foo.prototype, "fib") || _temp
if (_temp) Object.defineProperty(M.prototype, "fib", _temp)
return M
})();
```

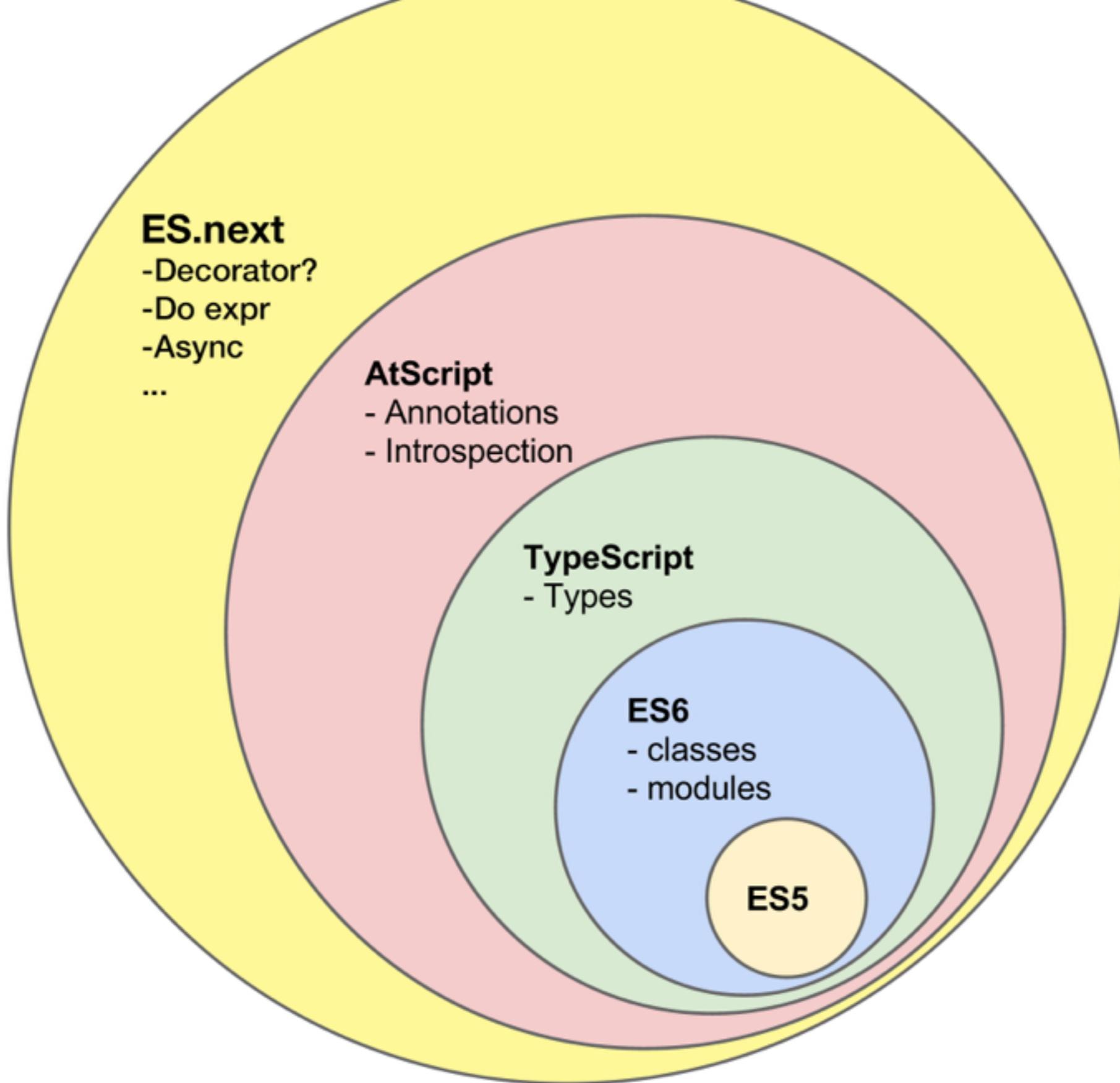
```
function memorize(target, name, descriptor) {
  let getter = descriptor.get, setter = descriptor.set;

  descriptor.get = function() {
    let table = memorizationFor(this);
    if (name in table) { return table[name]; }
    return table[name] = getter.call(this);
  }

  descriptor.set = function(val) {
    let table = memorizationFor(this);
    setter.call(this, val);
    table[name] = val;
  }

  return descriptor;
}
```





jonathandturner/brainstorming x

GitHub, Inc. [US] <https://github.com/jonathandturner/brainstorming>

GitHub This repository Search Explore Features Enterprise Blog Sign up Sign in

jonathandturner / brainstorming Watch 8 Star 11 Fork 2

Brainstorming

14 commits 1 branch 0 releases 2 contributors

branch: master brainstorming / +

Update decorators_and_metadata.md

jonathandturner authored 12 days ago latest commit a00e885745

README.md	spelling correction paramterTypes -> parameterTypes	14 days ago
decorators_and_metadata.md	Update decorators_and_metadata.md	12 days ago
syntax.md	Update syntax.md	2 months ago

README.md

1 Motivating examples:

<> Code

Issues 0

Pull requests 0

Pulse

Graphs

HTTPS clone URL

<https://github.com>

You can clone with [HTTPS](#) or [Subversion](#).

Clone in Desktop

Download ZIP

<https://github.com/jonathandturner/brainstorming>

- **Another version by Jonathan Turner**
- **Work for TypeScript at Microsoft**
- **TC39 member, work for decorator now**

Questions?