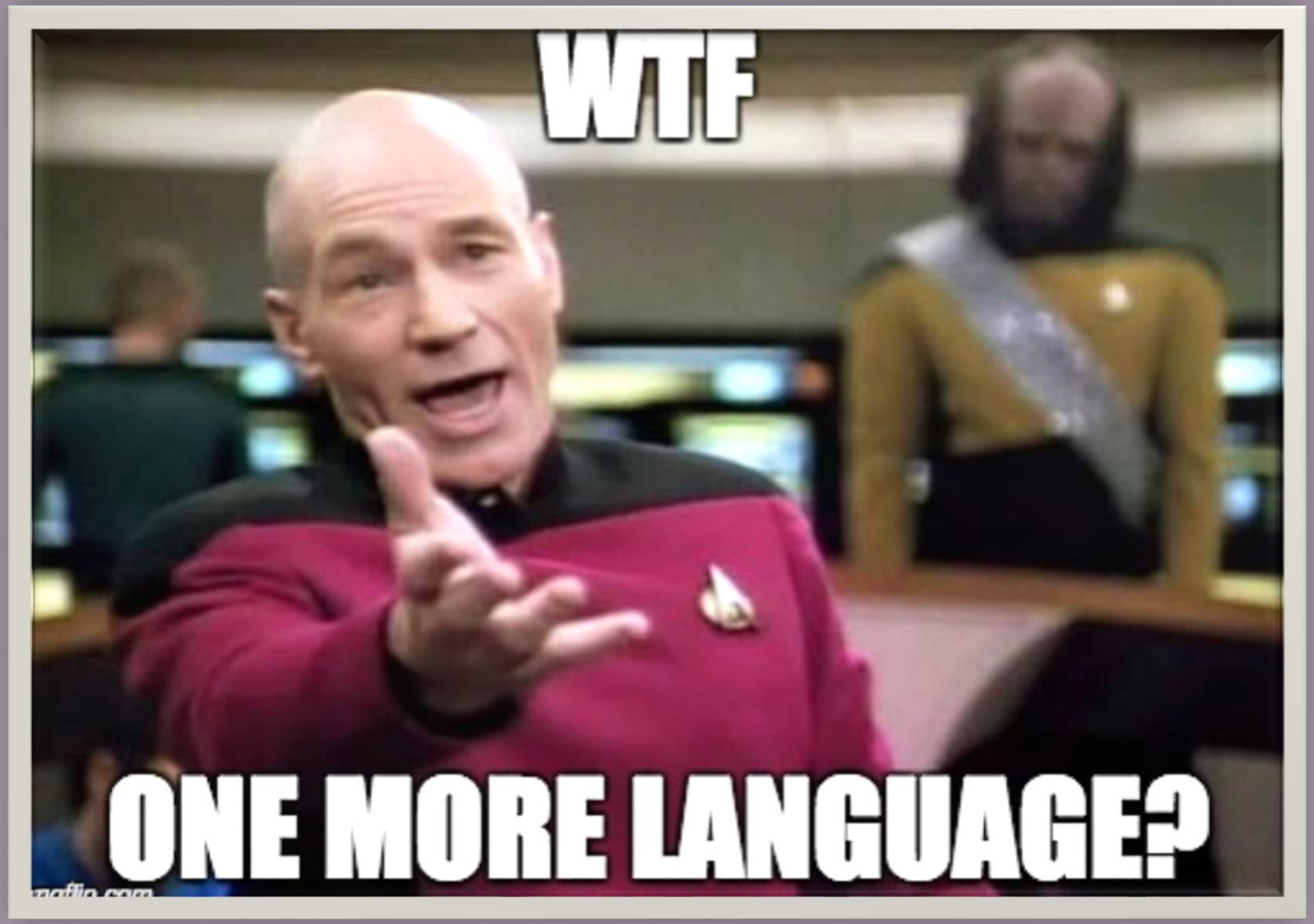
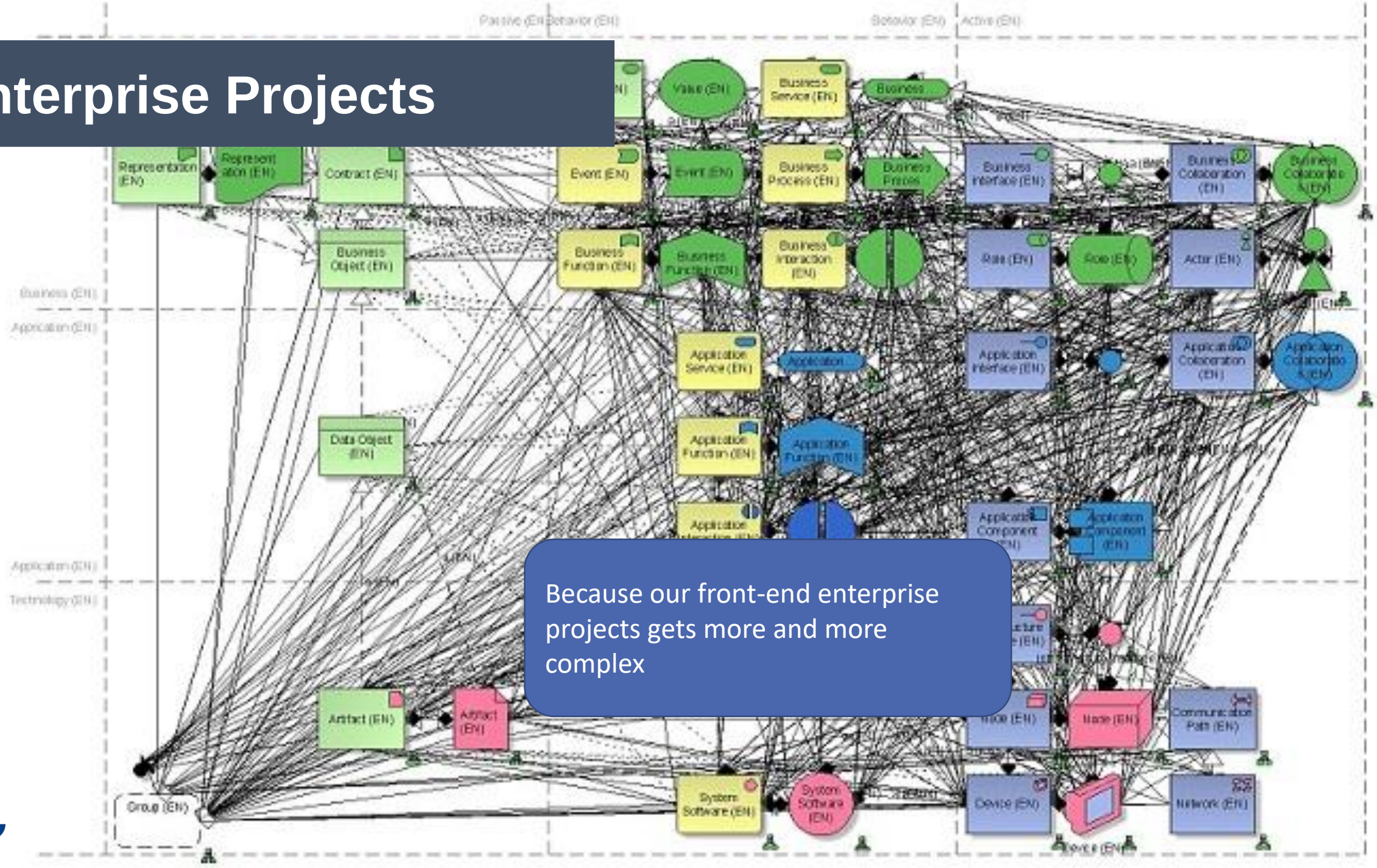
Typescript

is a language developed by

Microsoft


WTF
ONE MORE LANGUAGE?

Enterprise Projects



Atwood's Law

Any application that *can* be written in JavaScript, *will* eventually be written in JavaScript.

http://www.codinghorror.com/blog/2007/07/the-principle-of-least-power.html

JavaScript was originally developed for applications with a few hundred lines of code!

"Netscape", 1995


I JUST LET SOMEBODY ELSE
38
MAINTAIN MY JAVASCRIPT

TypeScript

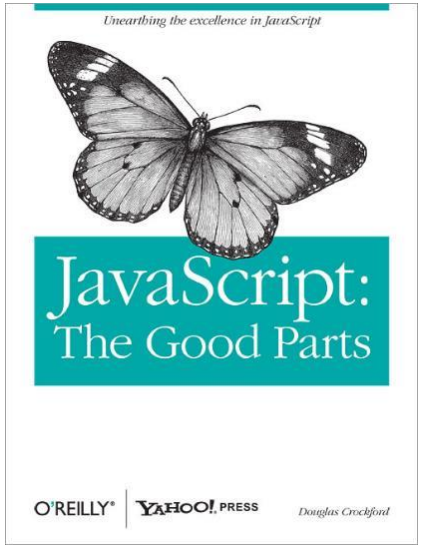
A typed superset of JavaScript that compiles to plain JavaScript

Any browser
Any host
Any OS

Open Source

URL: https://github.com/Microsoft/TypeScript

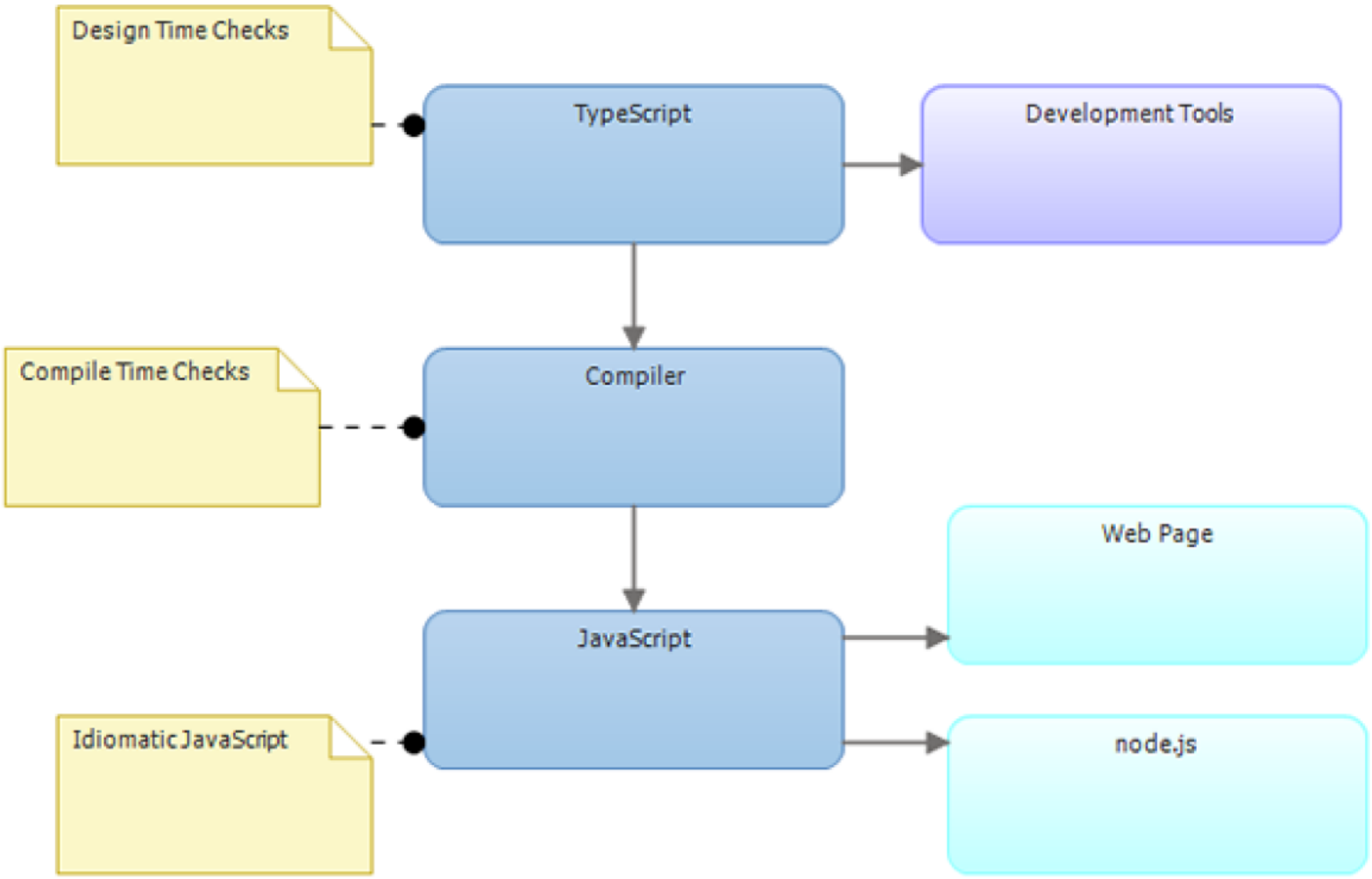
TypeScript is designed to supplement and improve JavaScript by providing the missing features and great tooling



+

+

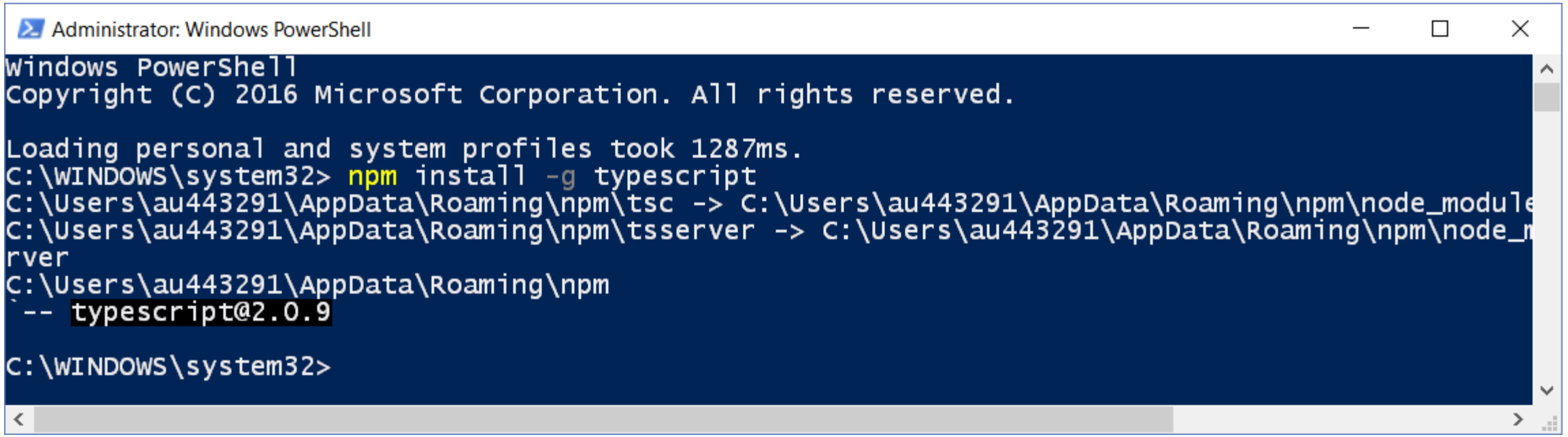
TypeScript life cycle



How to install?

- For NPM users:

```
npm i -g typescript
```

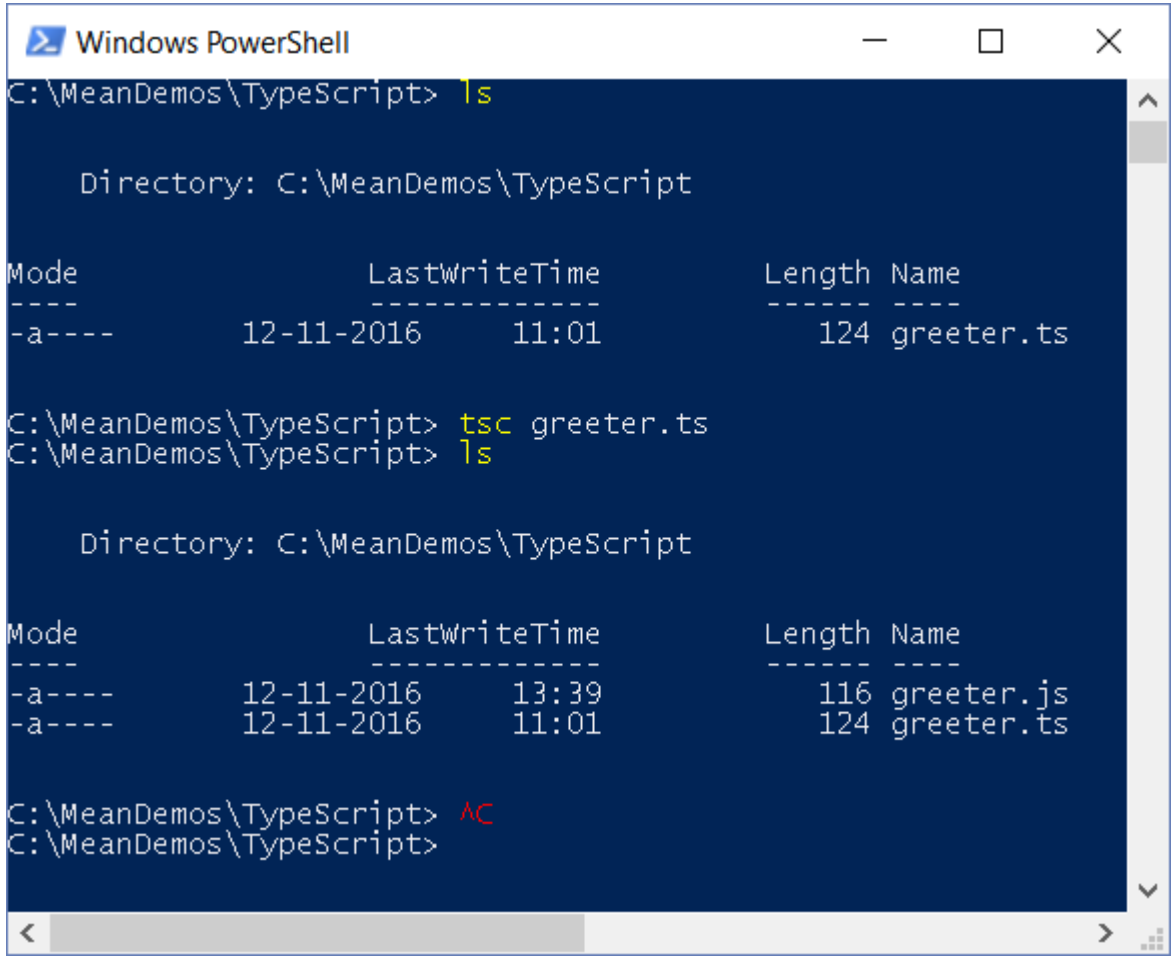


Compiling

- Run the TypeScript compiler:

```
tsc fileName.ts
```

  - You may need to restart your shell to update the PATH environment variable



Type System

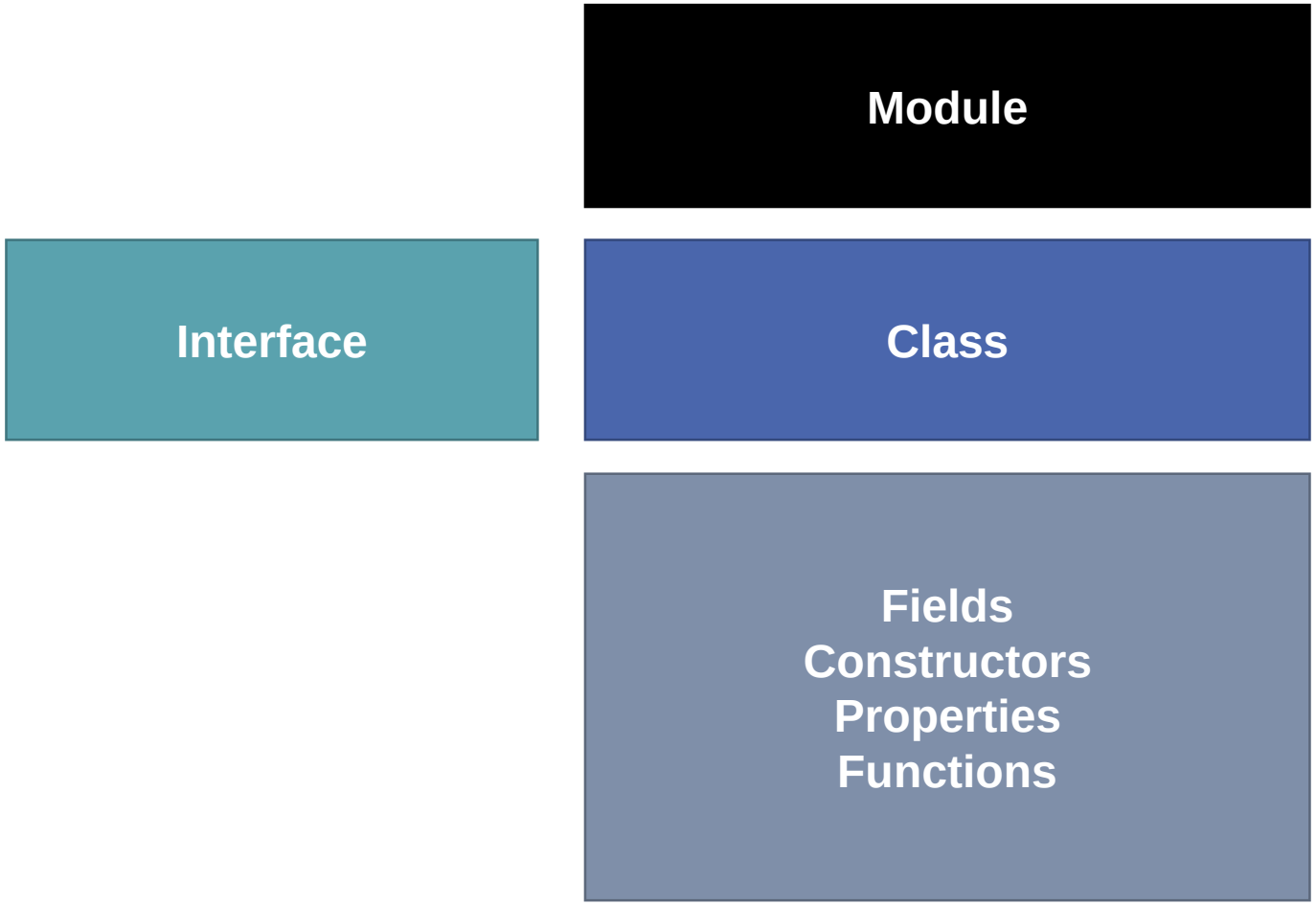
An accurate static representation of JavaScript's dynamic run-time type system

- Structural typing and type inference
  - In practice very few type annotations are necessary
- Generics
  - Increases accuracy and expressiveness of type system
- Works with existing JavaScript libraries
  - Declaration files can be written and maintained separately
- Types enable tooling
  - Provide verification and assistance, but not hard guarantees

Classes, Interfaces, Modules

- Scalable application structuring
  - Classes, interfaces, and modules enable clear contracts in code
- Supports popular module systems
  - CommonJS, AMD modules and ES modules

Code Hierarchy



Type annotations

TypeScript:

```
function greeter(person: string) {
    return "Hello, " + person;
}

var user = "John Doe";

console.log(greeter(user));
```

JavaScript:

```
function greeter(person) {
   return "Hello, " + person;
}
var user = "John Doe";
console.log(greeter(user));
```

Classes & Interfaces

TypeScript:

```
class Student {
    fullName: string;
    constructor(public firstName, public middleInitial,
public lastName) {
        this.fullName = firstName + " " + middleInitial
+ " " + lastName;
    }
}

interface Person {
    firstName: string;
    lastName: string;
}

function greeter(person : Person) {
    return "Hello, " + person.firstName + " " +
person.lastName;
}

var user = new Student("John", "M.", "Doe");
console.log(greeter(user));
```

JavaScript:

```
var Student = (function () {
   function Student(firstName, middleInitial, lastName) {
      this.firstName = firstName;
      this.middleInitial = middleInitial;
      this.lastName = lastName;
      this.fullName = firstName + " " + middleInitial + " " +
lastName;
   }
   return Student;
}());
function greeter(person) {
   return "Hello, " + person.firstName + " " +
person.lastName;
}
var user = new Student("John", "M.", "Doe");
console.log(greeter(user));
```

Decorators

- A Decorator is a special kind of declaration that can be attached to a class declaration, method, accessor, property, or parameter
- Decorators use the form @expression, where expression must evaluate to a function that will be called at runtime with information about the decorated declaration

Decorator example

TypeScript:

```
function ClassDecoratorParams(param: string) {
    return function(
        target: Function // The class the decorator is
declared on
        ) {
        console.log("ClassDecoratorParams(" +
         param + ") called on: ", target);
    }
}

@ClassDecoratorParams("a")
@ClassDecoratorParams("b")
class ClassDecoratorParamsExample {
}
```

JavaScript:

```
var __decorate = (this && this.__decorate) || function (deco
   var c = arguments.length, r = c < 3 ? target : desc === null ?
   if (typeof Reflect === "object" && typeof Reflect.decorate
   else for (var i = decorators.length - 1; i >= 0; i--) if (d = deco
   return c > 3 && r && Object.defineProperty(target, key, r),
};
function ClassDecoratorParams(param) {
   return function (target // The class the decorator is declare
      ) {
      console.log("ClassDecoratorParams(" +
         param + ") called on: ", target);
   };
}
var ClassDecoratorParamsExample = (function () {
   function ClassDecoratorParamsExample() {
   }
   ClassDecoratorParamsExample = __decorate([
      ClassDecoratorParams("a"),
      ClassDecoratorParams("b")
   ], ClassDecoratorParamsExample);
   return ClassDecoratorParamsExample;
```

How to install types?

- Some npm modules come with the types included when you install them.

- If the type file is missing you can install it manually with the command

```
> npm i @types/some-npm-package -D
```

  Example:

  Install the Express framework

```
> npm i express -S
```

  Install the TypeScript types for the Express framework as a development dependency:

```
> npm i @types/express -D
```

- For additional info see DefinitelyTyped
  - The repository for high quality TypeScript type definitions
  - http://definitelytyped.org/

References & Links

- Anders Hejlsberg about TypeScript 2
  https://channel9.msdn.com/Blogs/Seth-Juarez/Anders-Hejlsberg-on-TypeScript-2

- TypeScript tutorial
  http://www.typescriptlang.org/docs/tutorial.html

- Reference
  http://www.typescriptlang.org/docs/home.html