



JavaScript can feel messy!

A modern bedroom with deep purple walls and a high ceiling. A wooden bed with a white duvet and pillows is positioned against the wall. To the left of the bed is a wooden nightstand with a modern lamp. Two framed pictures hang on the wall above the bed. A large window with wooden blinds is on the right, offering a view of a green landscape. A television is mounted on the wall to the right of the window. The floor is made of polished wood.

We want maintainable code

JavaScript Code Encapsulation



Function Spaghetti Code



Ravioli Code
(JavaScript Patterns)

JavaScript Dynamic Types

- **JavaScript provides a dynamic type system**
- **The Good:**
 - Variables can hold any object
 - Types determined on the fly
 - Implicit type coercion (ex: string to number)
- **The Bad:**
 - Difficult to ensure proper types are passed without tests
 - Not all developers use ===
 - Enterprise-scale apps can have 1000s of lines of code to maintain

Migrating from Server-Side to Client-Side

- Migrating from server-side apps to client-side apps can be challenging



What are the Alternatives?

- **Several TypeScript alternatives exist:**
 - Write pure JavaScript
 - Apply JavaScript patterns
 - CoffeeScript – <http://coffeescript.org>
 - Dart – <http://dartlang.org>

TypeScript



DART



Shouldn't we Simply Write Plain JavaScript?



C \ C++

```
START: JUMP    LOOP      # jump past sensor and constant locations
RSV:   0000          # read right sensor value here
LSV:   0000          # read left sensor value here
RMP:   0000          # write right motor power level here
LMP:   0000          # write left motor power level here
OFF:   0000          # store motor-off constant here
ON:    0100          # store motor-on constant here
LOOP:  LOAD     1      RSV # load right sensor value into register 1
      LOAD     2      LSV # load left sensor value into register 2
      SUB      1  2    3 # subtract 1 from 2 and store result in 3
      LOAD     1      OFF # load motor-off constant into register 1
      LOAD     2      ON  # load motor-on constant into register 2
      BRANCH   3      RGT # if the left sensor is greater than the
LFT:   STORE    2      RMP # right then turn the right motor on
      STORE    1      LMP # and turn the left motor off
      JUMP     LOOP    # and then jump to beginning of the loop
```


What is TypeScript?

"TypeScript is a typed superset of JavaScript that compiles to plain JavaScript." ~ typescriptlang.org



Flexible Options

Any Browser

Any Host

Any OS

Open Source

Tool Support

Key TypeScript Features



Supports
standard
JavaScript
code

Provides
static typing

Encapsulation
through classes
and modules

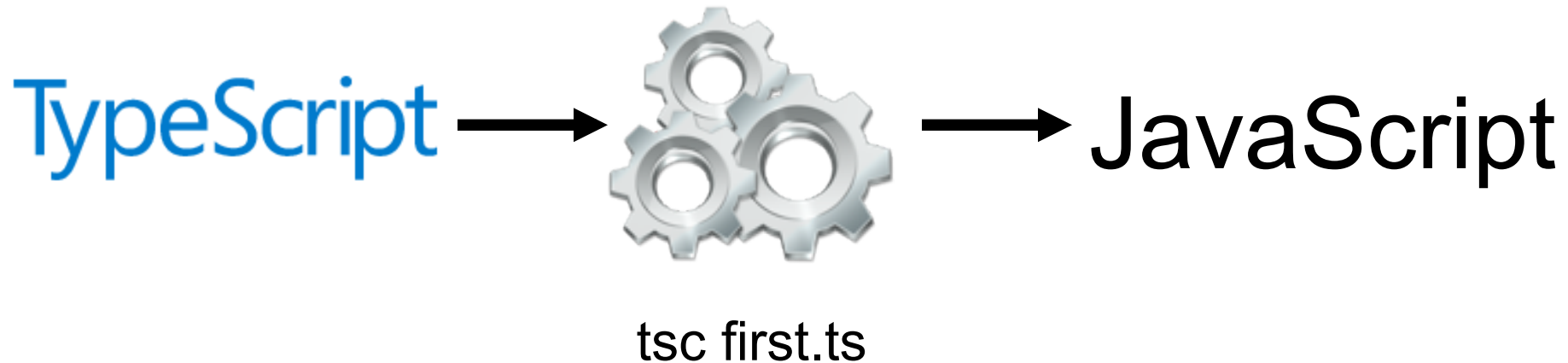
Support for
constructors,
properties,
functions

Define
interfaces

=> function
support
(lambdas)

Intellisense
and syntax
checking

TypeScript Compiler



TypeScript → JavaScript

TypeScript

Encapsulation

```
class Greeter {  
  greeting: string;  
  constructor (message: string) {  
    this.greeting = message;  
  }  
  greet() {  
    return "Hello, " + this.greeting;  
  }  
}
```

Static Typing

JavaScript

```
var Greeter = (function () {  
  function Greeter(message) {  
    this.greeting = message;  
  }  
  Greeter.prototype.greet = function () {  
    return "Hello, " + this.greeting;  
  };  
  return Greeter;  
})();
```

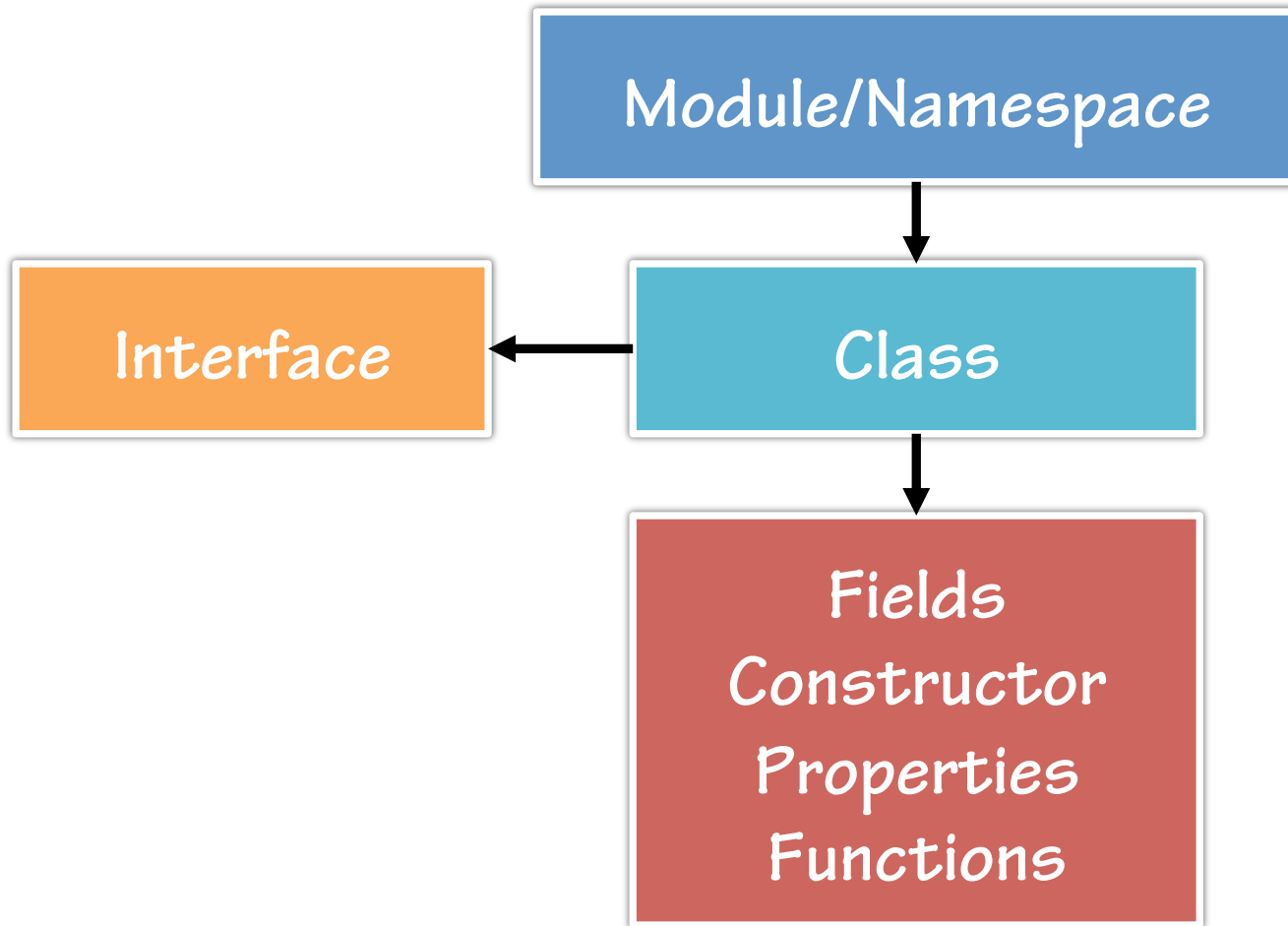

TypeScript Syntax Rules

- **TypeScript is a superset of JavaScript**
- **Follows the same syntax rules:**
 - { } brackets define code blocks
 - Semi-colons end code expressions
- **JavaScript keywords:**
 - for
 - if
 - More..

Important Keywords and Operators

Keyword	Description
class	Container for members such as properties and functions
constructor	Provides initialization functionality in a class
exports	Export a member from a module
extends	Extend a class or interface
implements	Implement an interface
imports	Import a module
interface	Defines code contract that can be implemented by types
module / namespace	Container for classes and other code
public/private	Member visibility modifiers
...	Rest parameter syntax
=>	Arrow syntax used with definitions and functions
<typeName>	< > characters use to cast/convert between types
:	Separator between variable/parameter names and types

Code Hierarchy



Tool/Framework Support

Node.js

Sublime

Emacs

Vi

Visual Studio

TypeScript Playground

TypeScript

PREVIEW

learn

play

get it

run it

join in

TypeScript

Walkthrough: Classes



Run

JavaScript

```
1
2 class Greeter
3 {
4     greeting: string;
5     constructor (message: string)
6     {
7         this.greeting = message;
8     }
9     greet()
10    {
11        return "Hello, " + this.greeting;
12    }
13 }
14
15 var greeter = new Greeter("world");
16
17 var button = document.createElement('button')
18 button.innerText = "Say Hello"
19 button.onclick = function() {
20     alert(greeter.greet())
21 }
```

```
1 var Greeter = (function () {
2     function Greeter(message) {
3         this.greeting = message;
4     }
5     Greeter.prototype.greet = function () {
6         return "Hello, " + this.greeting;
7     };
8     return Greeter;
9 })();
10 var greeter = new Greeter("world");
11 var button = document.createElement('button');
12 button.innerText = "Say Hello";
13 button.onclick = function () {
14     alert(greeter.greet());
15 };
16 document.body.appendChild(button);
17
```

Summary

- **TypeScript is an open source language that compiles to JavaScript**
- **Key features:**
 - Code encapsulation
 - Type support
- **Supports multiple tools:**
 - Node.js
 - Sublime (and others)
 - Visual Studio