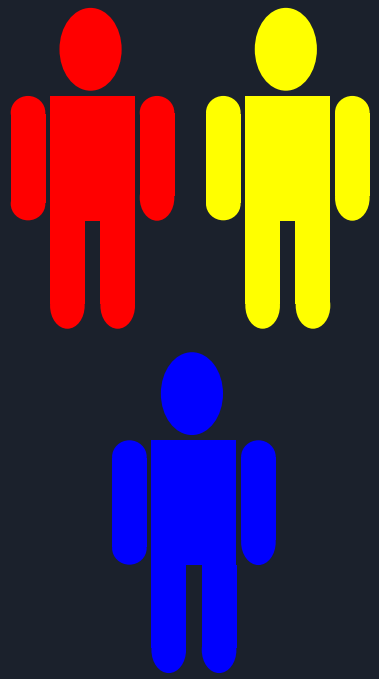
Scaling JavaScript Applications with TypeScript

Ryan Waskiewicz

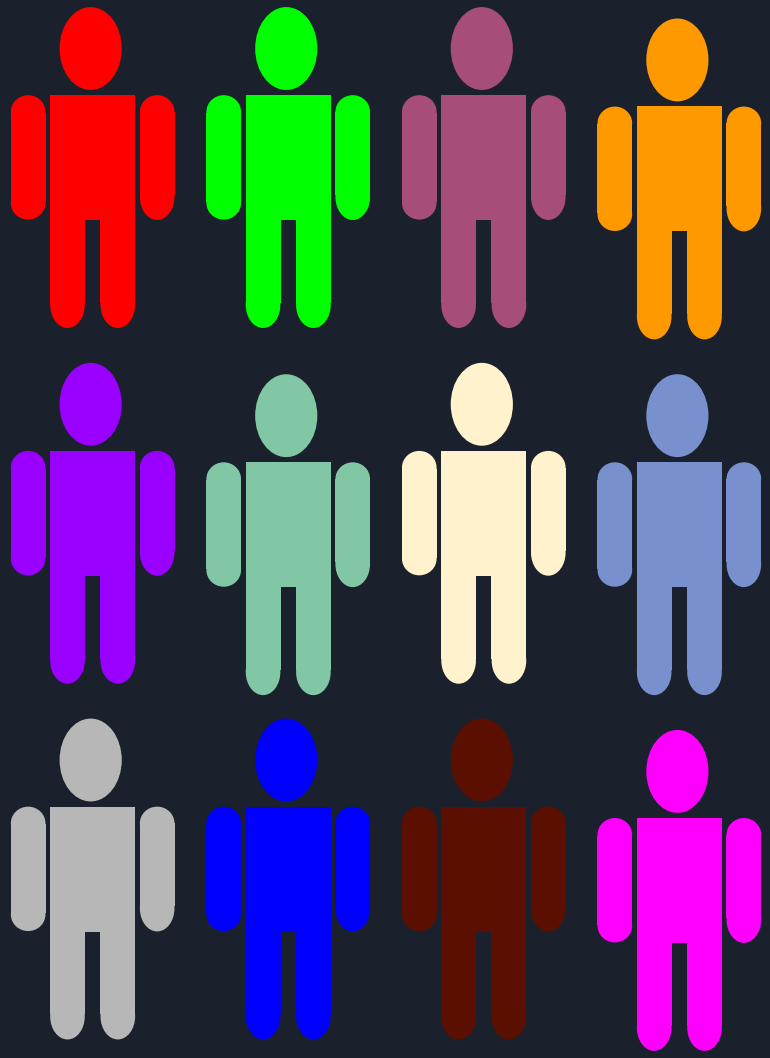
Web Applications for Self Driving Cars!

What is ‘Scalability’?

What is ‘Scalability’?

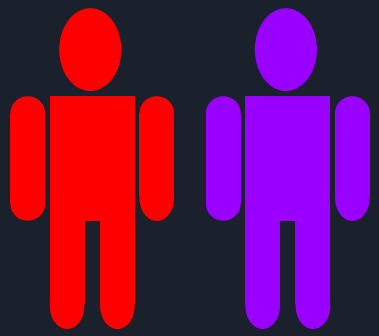


January 2018

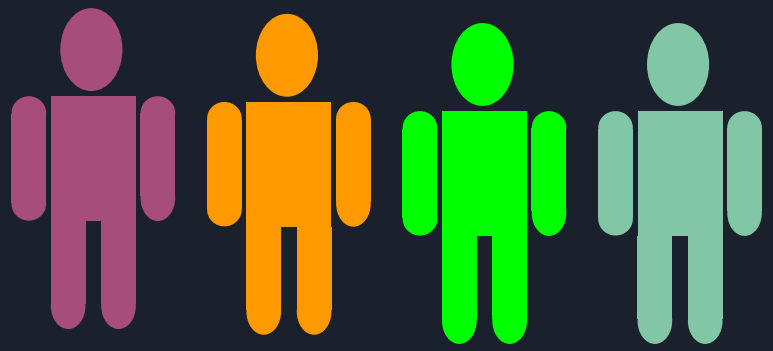


June 2018

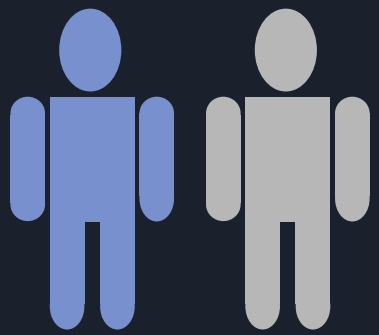
Pockets of Information



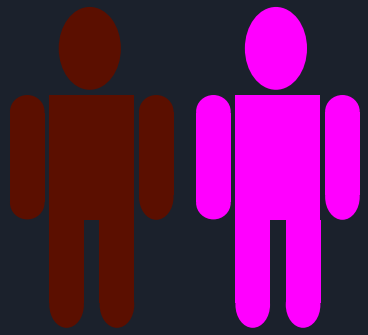
Authentication



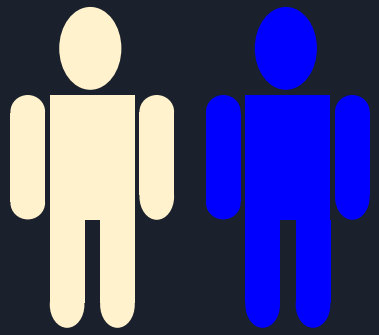
HTML Drawing



Infrastructure



Architecture



APIs

N(N-1)/2 = 10 Lines of Communication!

Code: Setting the State of a Traffic Bulb

```
function turnBulbOn(bulb) {
  bulb.isLit = true;
}
```

Team A: `boolean` representation

```
function toggleBulbState(bulb) {
  bulb.isLit = bulb.isLit ? 0 : 1;
}
```

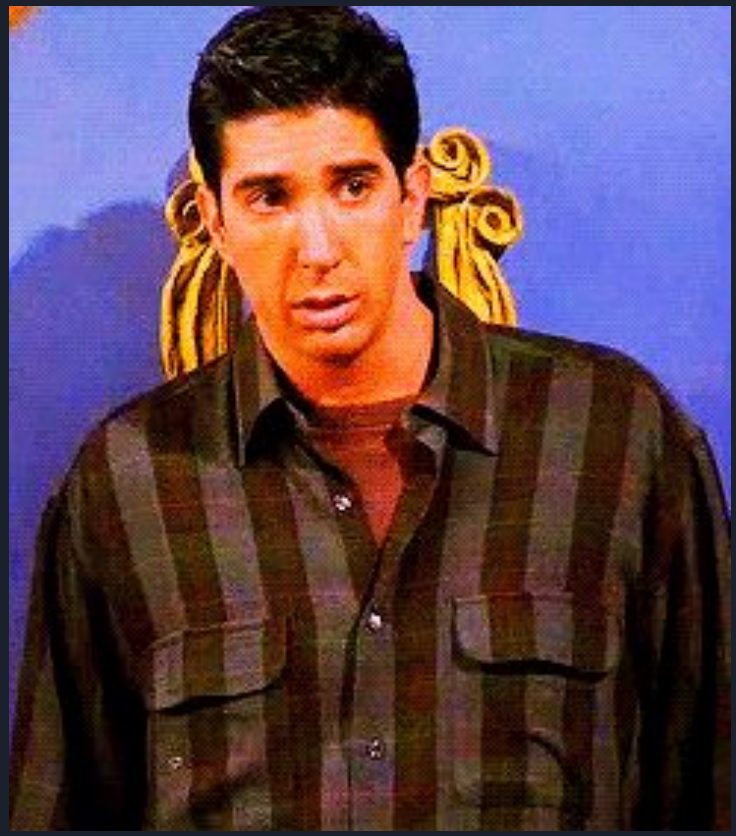
Team B: `number` representation

```
const bulbStates = Object.freeze({
  ON: 'on',
  OFF: 'off',
});

function turnBulbOn(bulb) {
  bulb.isLit = bulbStates.ON;
}
```

Team C: `enum` representation

Where Does this Fragmentation Lead Us?

```
const bulbStates = Object.freeze({
  ON: 'on',
  OFF: 'off',
});

function turnBulbOn(bulb) {
  bulb.isLit = true;
}

function isBulbOn(bulb) {
  return bulb.isLit === bulbStates.ON;
}

const bulb = new Bulb();
turnBulbOn(bulb);
console.log(`Bulb state is ${isBulbOn(bulb)}`);
// Bulb state is false
```

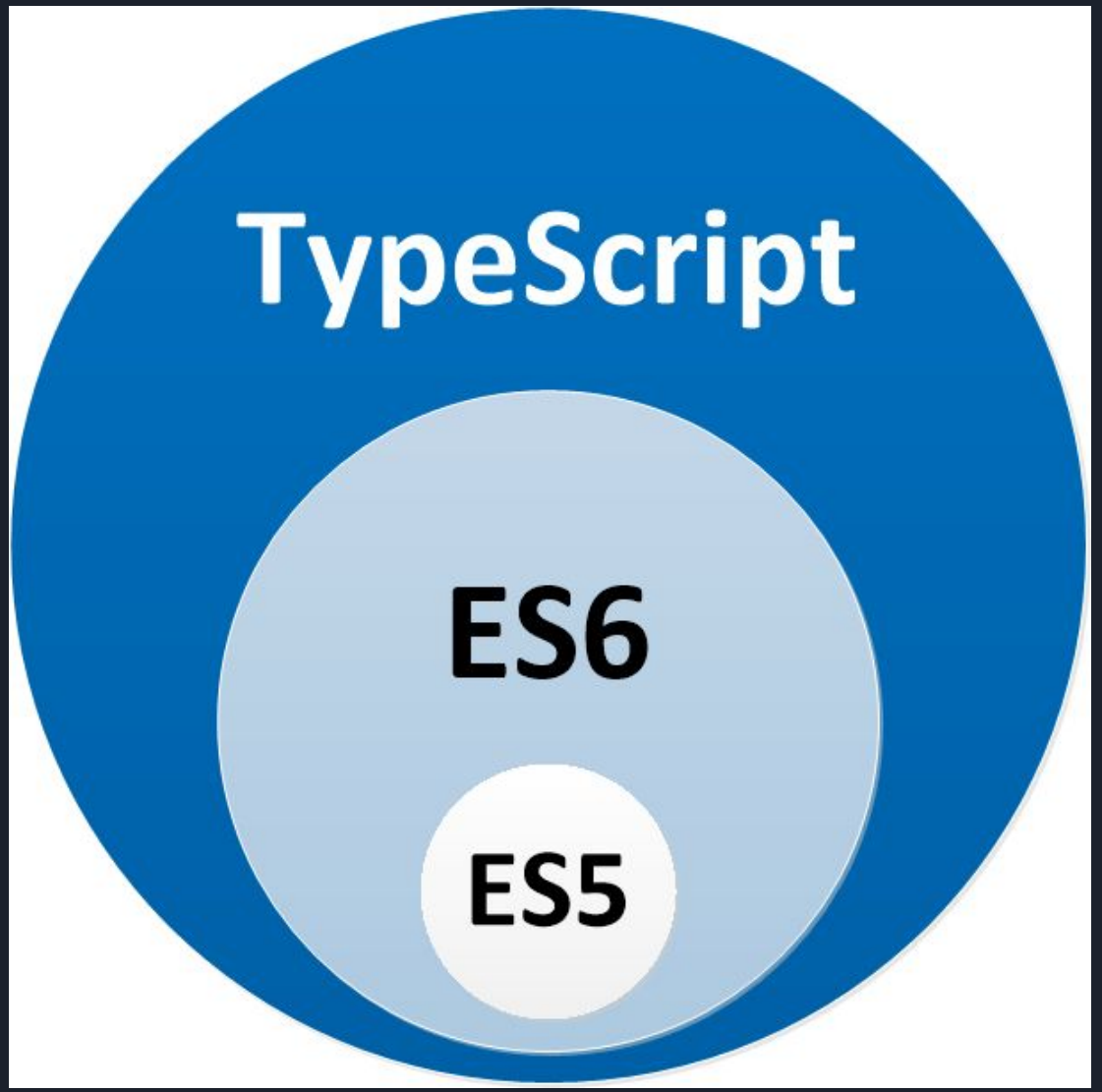


What Just Happened?

```
const bulbStates = Object.freeze({
  ON: 'on',
  OFF: 'off',
});

function turnBulbOn(bulb) {
  bulb.isLit = true;
}

function isBulbOn(bulb) {
  return bulb.isLit === bulbStates.ON;
}

const bulb = new Bulb();
turnBulbOn(bulb);
console.log(`Bulb state is ${isBulbOn(bulb)}`);
// Bulb state is false
```

- We tried to reuse code that was (presumably) tested
- There are two sources of truth, who's correct?
- Should we go back and 'fix' the 'wrong' code?

TypeScript



Static Typing

JavaScript itself has six primitive types

- *boolean, number, string, null, undefined, Symbol*

TypeScript is a *statically typed* language

Declare the *type* that variables will have within our code

```
class Bulb {
  constructor(color, isLit) {
    this.color = color;
    this.isLit = isLit;
  }

  setIsLit(newLitValue) {
    this.isLit = newLitValue;
  }
}
```

```
class Bulb {
  constructor(public color: string, public isLit: boolean) {
    this.color = color;
    this.isLit = isLit;
  }

  setIsLit(newLitValue: boolean) {
    this.isLit = newLitValue;
  }
}
    this.isLit = newLitValue;
  }
}
```

So...How Does My Code Run in the Browser?

```ts
class Bulb {
  constructor(public color: string, public isLit: boolean) {
    this.color = color;
    this.isLit = isLit;
  }

  setIsLit(newLitValue: boolean) {
    this.isLit = newLitValue;
  }
}
```

TS Compiler

tsconfig.json

```ts
class Bulb {
  constructor(color, isLit) {
    this.color = color;
    this.isLit = isLit;
  }

  setIsLit(newLitValue) {
    this.isLit = newLitValue;
  }
}
```

How Does Static Typing Benefit Us?

```
class Bulb {
  constructor(public color: string, public isLit: boolean) {
    this.color = color;
    this.isLit = isLit;
  }

  setIsLit(newLitValue: boolean) {
    this.isLit = newLitValue;
  }
}
```

```
// Argument of type '0' is not assignable to parameter of type 'boolean'.
const redBulb = new Bulb('red', 0);
// Argument of type 'false' is not assignable to parameter of type 'string'.
const yellowBulb = new Bulb(false, 'yellow');
// Expected 2 arguments, but got 3.
const greenBulb = new Bulb('green', true, 'left arrow');
// Correct types, used in the correct order :-)
const workingBulb = new Bulb('green', true);
```

Finding Bugs through Conversion

```
1   class Bulb {
2     constructor(public color: string, public isLit: boolean) {
3       this.color = color;
4       this.isLit = isLit;
5     }
6
7     setIsLit(newLitValue: boolean) {
8       this.isLit = newLitValue;
9     }
10  }
11
12  class TrafficLight {
13    constructor(private bulbs: Bulb[]) {
14      this.bulbs = bulbs;
15    }
16
17    setLitBulbByColor(colorToLight: string) {
18      this.bulbs.forEach((bulb) => {
19        bulb.isLit = (colorToLight === bulb.colour);
20      });
21    }
22  }
23
24  const light = new TrafficLight(['red', 'yellow', 'green']);
25  light.setLitBulbByColor('red');
26
```

Adopting TypeScript: Should We Do This?

1. Is there team interest?
2. Can we get buy-in?
3. Technologically speaking, could we do this?

When *Doesn't* TypeScript Make Sense

- Framework support/lack of examples
- Infrastructure is too hard to migrate
- Your Team May Want to Look at Alternatives

Adopting TypeScript: Let's Tackle It!

1. Start Small - Convert 1 JS file to TS
    a. Right Click -> Rename file -> yourFile.ts
2. Let's update our build file to compile TS files
    a. Generate tsconfig file
        i. `npm install -g typescript`
        ii. `tsc --init`
    b. Integrating TS w/Build Systems: https://bit.ly/2J5n2AZ
3. Build!
4. Fix Type Errors
5. Accept this may be a gradual process
6. Rinse, Repeat
7. Linting + Formatting

Lessons Learned Along the Way

1. We didn’t know everything at first - that’s OK!
2. Standardizing our builds really helped attain consistency
3. There’s always room for improvement
4. Gradualism is key!

Thank You!

Slides - https://bit.ly/2xcmCY5

TypeScript - https://www.typescriptlang.org/

Integrating TS w/Build Tools - https://bit.ly/2J5n2AZ

A Decade after DARPA Our View on the State of the Art in Self Driving Cars - https://bit.ly/2s6DgDg