REASON NYC

Wifi: `The Farm WiFi`

Password: `organicfarm`

Jared Palmer

@jaredpalmer

Agenda

- **Presentations**
- **Panel & Discussion**
- **Networking**

Conferences
Meetups
Books
Online forums
Blog posts
Videos
Tutorials
Examples
Documentation
Version control
Editor plugins
Comments
Tests
Code (language)

Conferences
Meetups
Books
Online forums
Blog posts
Videos
Tutorials
Examples
Documentation
Version control
Editor plugins
Comments
Tests
Code (language)

Conferences

Meetups

Books

Online forums

Blog posts

Videos

Tutorials

#goals

Examples

Documentation

Version control

Editor plugins

Comments

Tests

Code (language)

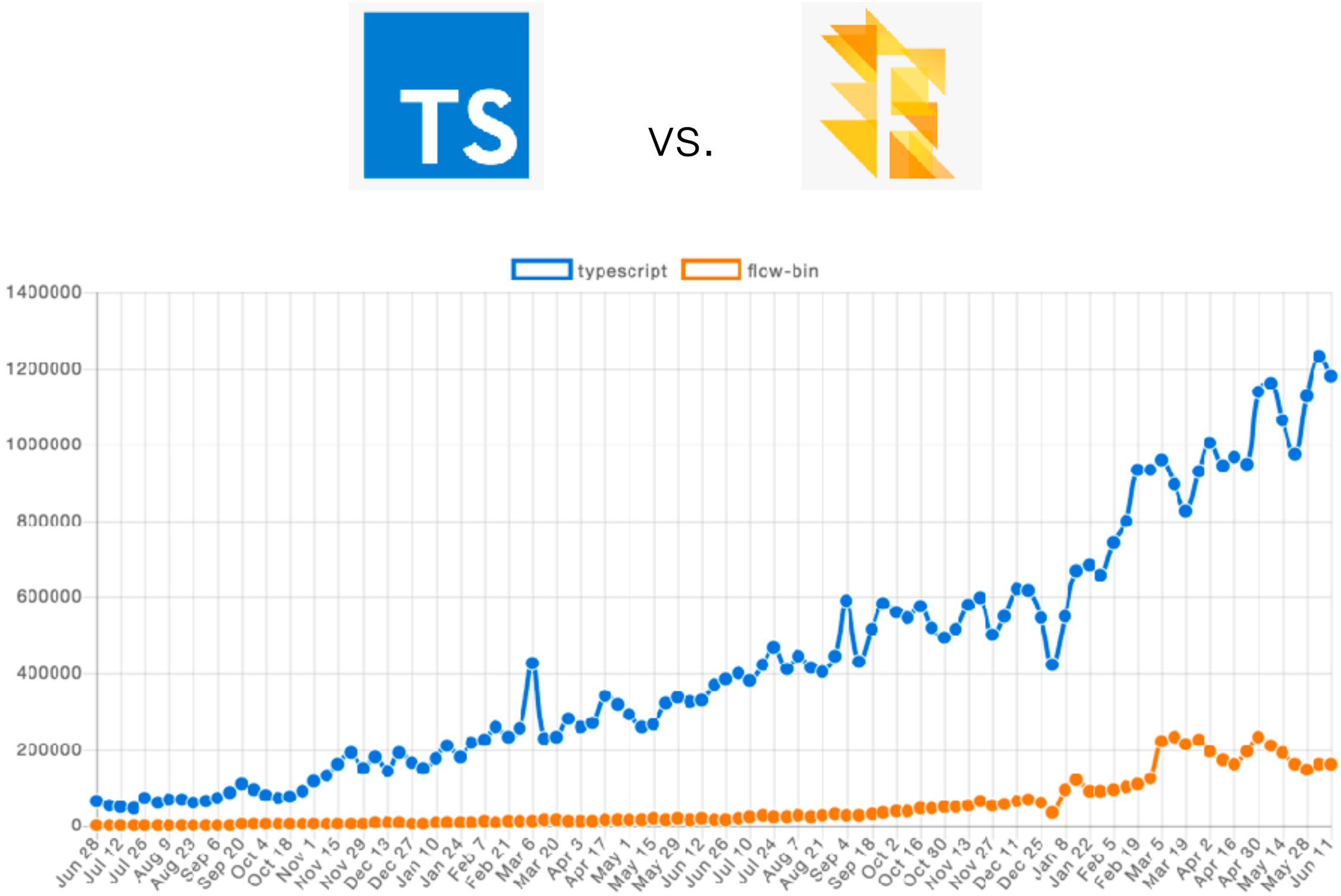
```
const petAnimal = (animal) => {
  switch (animal) {
    case "cat":
      // do something
      break;
    case "dog":
      // do something
      break;
  }
}

petAnimal(1) // ?
petAnimal("blah") // ?
petAnimal(() => "cat") // ?
```


BABEL

“Types, immutability, and pattern matching reduce accidental complexity”

@Sander_Spies


typescript
flow-bin
1400000
1200000
1000000
800000
600000
400000
200000
0
Jun 28
Jul 12
Jul 26
Aug 9
Aug 23
Sep 6
Sep 20
Oct 4
Oct 18
Nov 1
Nov 15
Nov 29
Dec 13
Dec 27
Jan 10
Jan 24
Feb 7
Feb 21
Mar 6
Mar 20
Apr 3
Apr 17
May 1
May 15
May 29
Jun 12
Jun 26
Jul 10
Jul 24
Aug 7
Aug 21
Sep 4
Sep 18
Oct 2
Oct 16
Oct 30
Nov 13
Nov 27
Dec 11
Dec 25
Jan 8
Jan 22
Feb 5
Feb 19
Mar 5
Mar 19
Apr 2
Apr 16
Apr 30
May 14
May 28
Jun 11

```ts
type Animal = 'cat' | 'dog' | 'bird';

const petAnimal = (animal: Animal) => {
  switch (animal) {
    case 'cat':
      // do something
      break;
    case 'dog':
      // do something
      break;
  }
};

petAnimal(1); // ?
petAnimal('blah'); // ?
petAnimal(() => 'cat'); // ?
```

```ts
type Animal = 'cat' | 'dog' | 'bird';

const petAnimal = (animal: Animal) => {
  switch (animal) {
    case 'cat':
      // do something
      break;
    case 'dog':
      // do something
      break;
  }
};

petAnimal(
petAnimal(
petAnimal(() => 'cat'); // ?
```

```
[ts]
Argument of type '() => string' is not assignable to parameter of
 type 'Animal'.
  Type '() => string' is not assignable to type '"bird"'.
```

“I think that JavaScript’s loose typing is one of its best features and that type checking is way overrated. TypeScript adds sweetness, but at a price. It is not a price I am willing to pay.”

– Doug Crockford, author of JavaScript, the Good Parts

- Types, immutability, pattern matching by default
- Compiler toolchain, JS/native/kernel
- Objects, classes, modules, language extensions, and more
- Functional or imperative code

OCaml's is not very "junior-dev friendly".

- “Meta-language” on top of OCaml
- Same compiler, more approachable syntax
- Comes with a “blessed” toolchain
- Extreme focus on developer experience
- Incrementally adoptable

Let Bindings and Scoping

```
let name = {
  let firstName ="jared";
  let lastName = "palmer";
  firstName ^ " " ^ lastName
};
/* name -> "jared palmer" */
```

The Type System

```
/* Type annotations when you want them */
string => string
let greet (name: string) :string => "Hello " ^ name;

/* Type inferences come for free */
string => string
let greet name => "Hello " ^ name;

/* Algebraic Data Types */
type point = (int, int);

/* Parametric polymorphism (generics) */
type point 'a = ('a, 'a);
```

Variants & Pattern Matching

```
type animal =
  | Cat
  | Dog
  | Bird;
```

```
animal => { }
```

```
let petAnimal animal =>
  switch animal {
  | Cat => {/* do something */}
  | Dog => {/* do something */}
};
```

Useful Error Messages

```
type animal =
  [merlin]
  Warning 8: this pattern-matching is not exhaustive.
  Here is an example of a value that is not matched:
  Bird
  | Dog
  | Bird;
  animal
    | Dog => {/* do something */}
  };
```

Functions

```
/*  Labeled arguments & Currying */
```

```
x::'a => y::'b => { }
```

```
let setCoordinates ::x ::y => {/* use x and y here */};

setCoordinates x::5 y::6;

setCoordinates x::20 y::100;
```

```
y::int => { }
```

```
let setY = setCoordinates x::10;

setY y::200;

setY y::300;
```

Functions

```
/* Pattern matching & Optional arguments */
color::'a => radius::'b? => unit => 'c
let drawCircle ::color ::radius=? () => {
  /* radius can be omitted */
  setColor color;
  switch radius {
  | None => startAt 1 1
  | Some r_ => startAt r_ r_
  }
};
```

Reason React

```
ReasonReact.componentSpec int ReasonReact.stateless
let component = ReasonReact.statefulComponent "Greeting";

name::string => 'a => ReasonReact.componentSpec int int
let make ::name _children => {
  let handleClick _event state _self => ReasonReact.Update (state + 1);
  {
    ...component,
    initialState: fun () => 0,
    render: fun state self => {
      let greeting = "Hello " ^ name ^ ". You've clicked " ^ (string_of_int state) ^ " time(s)!";
      <button onClick=(self.update handleClick)>
        (ReasonReact.stringToElement greeting)
      </button>
    }
  }
};
```

Other goodies

- Modules
- Mutable fields
- ref
- Imperative loops
- Typed comments
- refmt
- better-error

ML Convergence

Tooling and Config ⟶ ⟵ Language & Syntax

Is Reason ready?