Semmle™

How TypeScript is transforming the JavaScript ecosystem

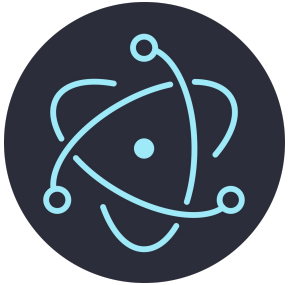
Sam Lanning

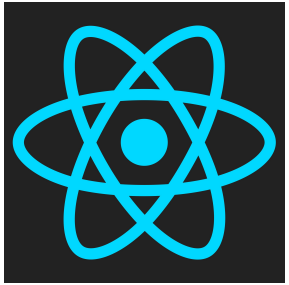
@samlanning @samlanning

About Me

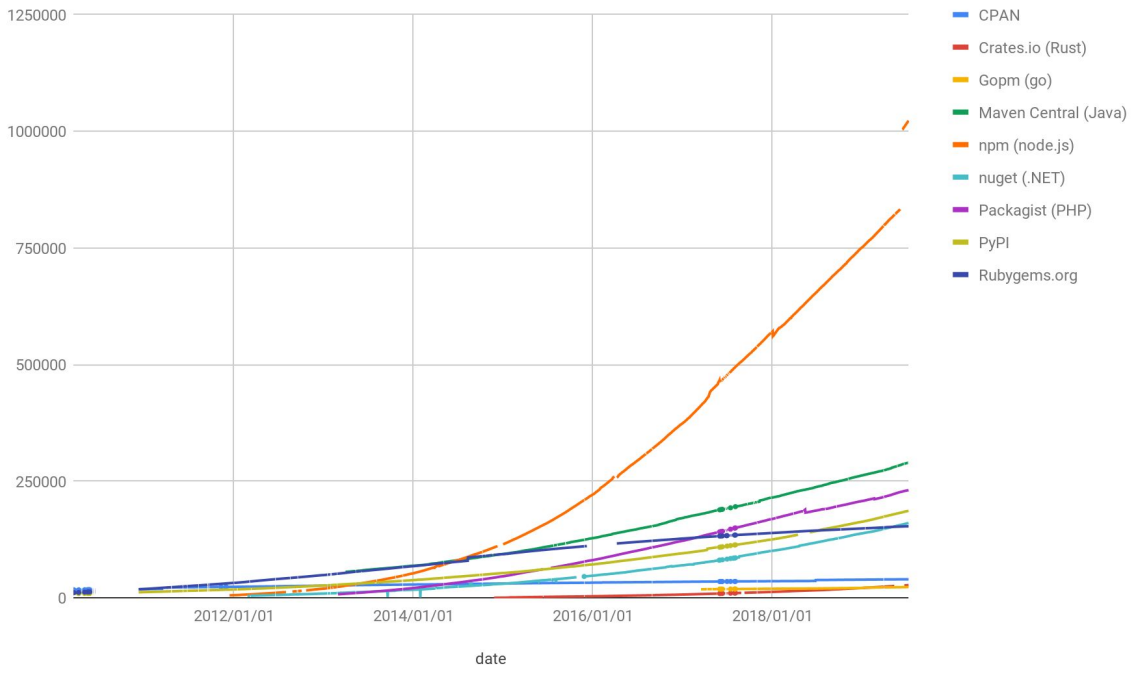
- Developer advocate for Semmle
  - (formerly core developer for LGTM.com)
- Been writing JavaScript for 15+ years
- Passionate about Open Source, Security, Cryptography, Code Quality, Lighting …
  … and TypeScript.
- Prefers **dark** themes to **light** themes
- Twitter/GitHub/NPM: @samlanning

JavaScript's popularity





JavaScript’s popularity



http://www.modulecounts.com/

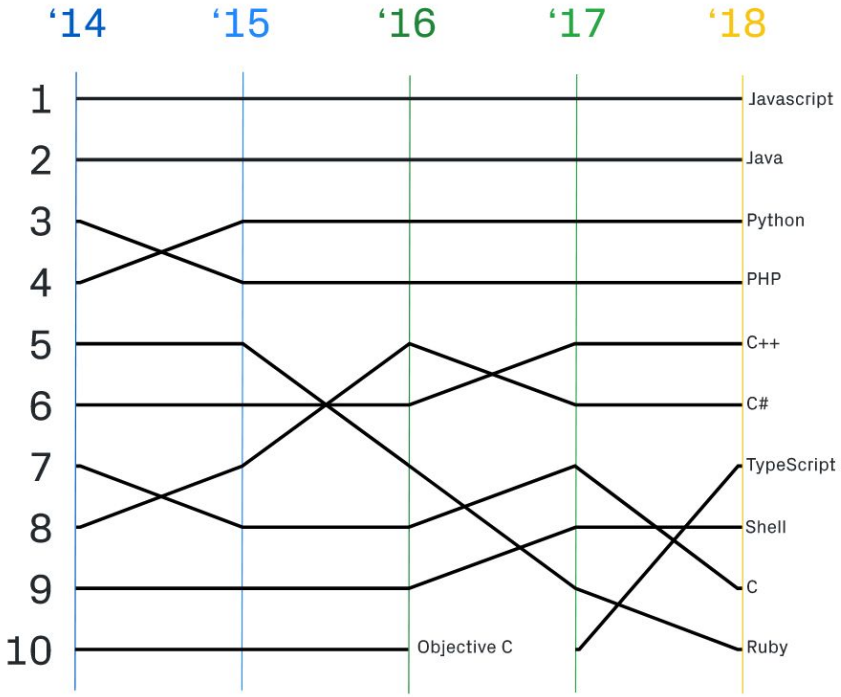
JavaScript's popularity

Top languages over time

You're coding on GitHub in hundreds of programming languages, but JavaScript still has the most contributors in public and private repositories, organizations of all sizes, and every region of the world.

This year, TypeScript shot up to #7 among top languages used on the platform overall, after making its way in the top 10 for the first time last year. TypeScript is now in the top 10 most used languages across all regions GitHub contributors come from—and across private, public, and open source repositories. *

Top 10 primary languages over time, ranked by number of unique contributors to public and private repositories tagged with the appropriate primary language.



https://octoverse.github.com/projects#languages

Large JavaScript Applications

broadcast.js

```javascript
export class Broadcaster {

  constructor() {
    this.listeners = [];
  }

  broadcastMessage(message) {
    this.listeners.forEach(l => l(message));
  }

  addListener(listener) {
    this.listeners.push(listener);
  }

}

```

Large JavaScript Applications

broadcast.js

```javascript
export class Broadcaster {

  constructor() {
    this.listeners = [];
  }

  broadcastMessage(message) {
    this.listeners.forEach(l => l(message));
  }

  addListener(listener) {
    this.listeners.push(listener);
  }

}

```

model.js

```javascript
import { Broadcaster } from '../../util/core/broadcast';

export const notifications = new Broadcaster();

// some stuff

notifications.broadcastMessage({
  foo: 'bar'
})

```

Large JavaScript Applications

broadcast.js

```javascript
export class Broadcaster {

  constructor() {
    this.listeners = [];
  }

  broadcastMessage(message) {
    this.listeners.forEach(l => l(message));
  }

  addListener(listener) {
    this.listeners.push(listener);
  }

}

```

model.js

```javascript
import { Broadcaster } from '../../util/core/broadcast';

export const notifications = new Broadcaster();

// some stuff

notifications.broadcastMessage({
  foo: 'bar'
})

```

receiver.js

```javascript
import { notifications } from '../../model/notifications/model'

notifications.addListener(m => {
  console.log(m.foo);
})

```

Large JavaScript Applications

broadcast.js

```javascript
export class Broadcaster {

  constructor() {
    this.listeners = [];
  }

  sendMessage(message) {
    this.listeners.forEach(l => l(message));
  }

  addListener(listener) {
    this.listeners.push(listener);
  }

}

```

model.js

```javascript
import { Broadcaster } from '../../util/core/broadcast';

export const notifications = new Broadcaster();

// some stuff

notifications.broadcastMessage({
  foo: 'bar'
})

```

receiver.js

```javascript
import { notifications } from '../../model/notifications/model'

notifications.addListener(m => {
  console.log(m.foo);
})

```

Large JavaScript Applications

broadcast.js

```javascript
export class Broadcaster {

  constructor() {
    this.listeners = [];
  }

  broadcastMessage(message) {
    this.listeners.forEach(l => l(message));
  }

  addListener(listener) {
    this.listeners.push(listener);
  }

}

```

model.js

```javascript
import { Broadcaster } from '../../util/core/broadcast';

export const notifications = new Broadcaster();

// some stuff

notifications.broadcastMessage({
  foo: 'bar'
})

// other stuff

notifications.broadcastMessage({
  goo: 'bar'
})

notifications.broadcastMessage()

```

Large JavaScript Applications

broadcast.js

```javascript
export class Broadcaster {

  constructor() {
    this.listeners = [];
  }

  broadcastMessage(message) {
    this.listeners.forEach(l => l(message));
  }

  addListener(listener) {
    this.listeners.push(listener);
  }

}

```

model.js

```javascript
import { Broadcaster } from '../../util/core/broadcast';

export const notifications = new Broadcaster();

// some stuff

notifications.broadcastMessage({
  foo: 'bar'
})

```

receiver.js

```javascript
import { notifications } from '../../model/notifications/model'

notifications.addListener(m => {
  console.log(m.goo)
})

```

Large JavaScript Applications


```
Uncaught (in promise) TypeError: undefined is not a function                VM52:2
    at <anonymous>:2:24
    at new Promise (<anonymous>)
    at <anonymous>:2:5
Uncaught (in promise) TypeError: Cannot read property 'foo' of undefined    VM52:3
    at <anonymous>:3:34
    at new Promise (<anonymous>)
    at <anonymous>:3:5
Uncaught (in promise) TypeError: Cannot read property 'foo' of null         VM52:4
    at <anonymous>:4:29
    at new Promise (<anonymous>)
    at <anonymous>:4:5
Uncaught (in promise) ReferenceError: foo is not defined                    VM52:5
    at <anonymous>:5:24
    at new Promise (<anonymous>)
    at <anonymous>:5:5
```

Large JavaScript Applications

"Types exist in JavaScript whether you choose to use tooling which can reason about them for you or not."

James Henry - Microsoft MVP for TypeScript

Static Typing for Javascript

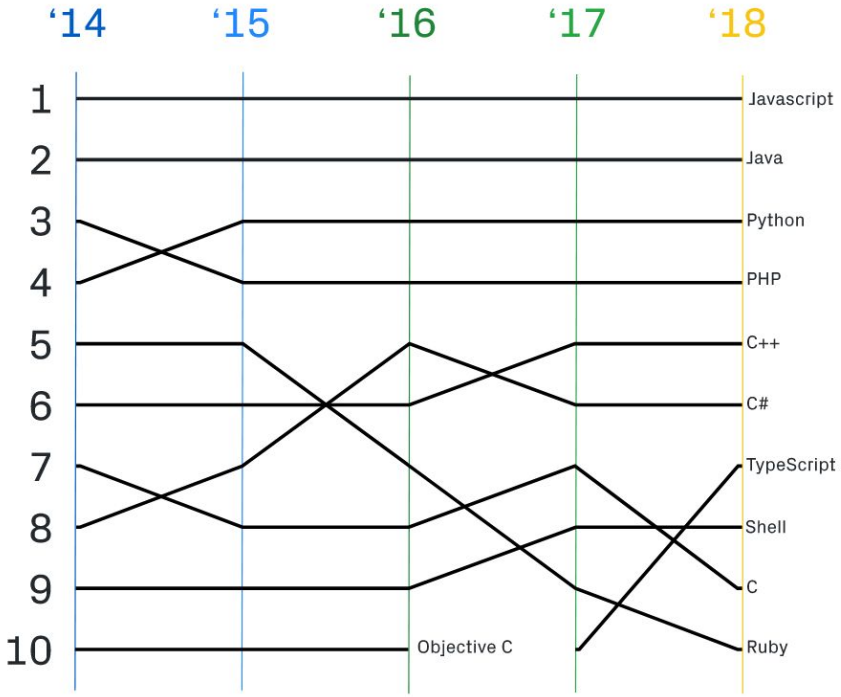
Static Typing for Javascript

Top languages over time

You're coding on GitHub in hundreds of programming languages, but JavaScript still has the most contributors in public and private repositories, organizations of all sizes, and every region of the world.

This year, TypeScript shot up to #7 among top languages used on the platform overall, after making its way in the top 10 for the first time last year. TypeScript is now in the top 10 most used languages across all regions GitHub contributors come from—and across private, public, and open source repositories. *

Top 10 primary languages over time, ranked by number of unique contributors to public and private repositories tagged with the appropriate primary language.



Introducing TypeScript

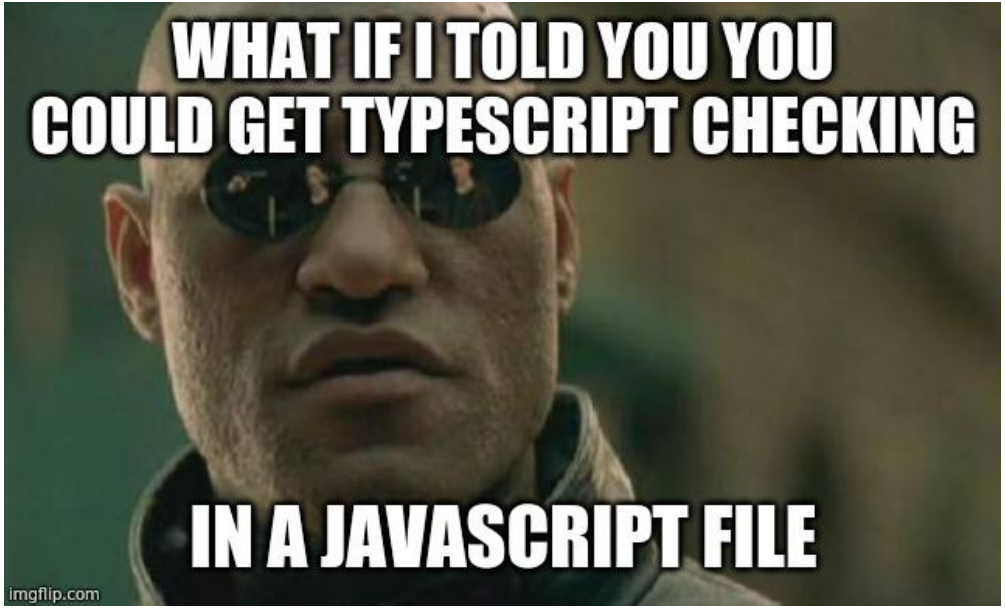
- Introduces static typing to JavaScript: *"JavaScript that scales"*
- TypeScript syntax is a **superset** of JavaScript
- TypeScript **does not** replace JavaScript
  - Made concessions to account for dynamic nature of JavaScript
  - Closely tracks the ECMAScript standard
- Lots and lots of tooling: IDE Integration with jump-to-definition, intellisense / auto-completion, error detection, refactoring etc...

TypeScript: State of the Union - James Henry - https://youtu.be/4nwb-kplv-k

// @ts-check

- or -

--checkJs



Introducing TypeScript

broadcast.ts

```typescript
export class Broadcaster<T> {

  private listeners: ((m: T) => void)[];

  constructor() {
    this.listeners = [];
  }

  public broadcastMessage(message: T) {
    this.listeners.forEach(l => l(message));
  }

  public addListener(listener: (m: T) => void) {
    this.listeners.push(listener);
  }

}

```

model.ts

```typescript
import { Broadcaster } from '../../util/core/broadcast';

export const notifications = new Broadcaster<{foo: string}>();

// some stuff

notifications.broadcastMessage({
  foo: 'bar'
})

```

receiver.ts

```typescript
import { notifications } from '../../model/notifications/model'

notifications.addListener(m => {
  console.log(m.foo);
})

```

Introducing TypeScript

broadcast.ts

```typescript
export class Broadcaster<T> {

  private listeners: ((m: T) => void)[];

  constructor() {
    this.listeners = [];
  }

  public broadcastMessage(message: T) {
    this.listeners.forEach(l => l(message));
  }

  public addListener(listener: (m: T) => void) {
    this.listeners.push(listener);
  }

}
```

model.ts

```typescript
import { Broadcaster } from '../../util/core/broadcast';

export const notifications = new Broadcaster<{foo: string}>();

// some stuff

notifications.broadcastMessage({
  foo: 'bar'
})
```

receiver.ts

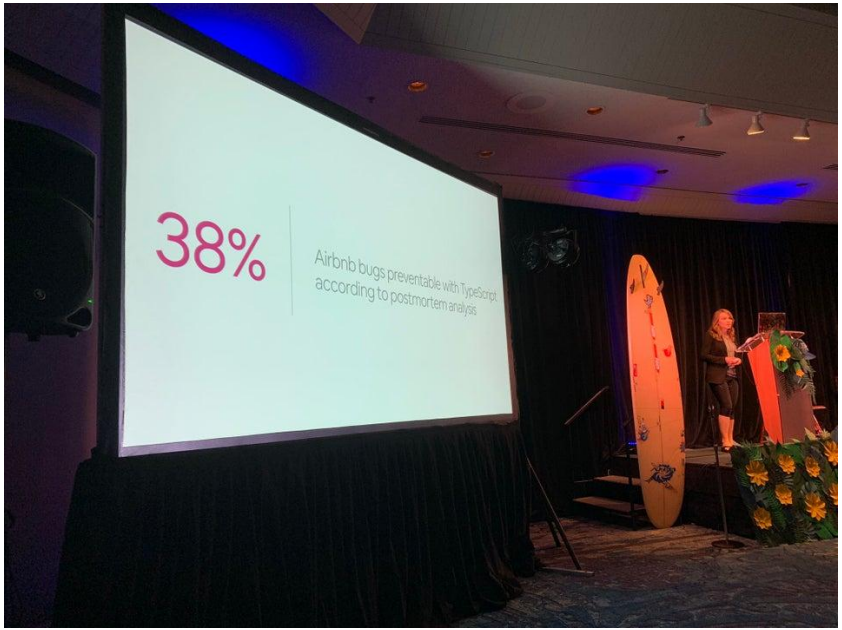
```typescript
import { notifications } from '../../model/notifications/model'

notifications.addListener(m => {
  console.log(m.foo);
})
```

Introducing TypeScript

Demo Time! (1/4)

Does it actually make a big difference?



Does it actually make a big difference?

lgtm.com

External NPM Modules

application.ts × | lodash.d.ts

typescript-talks ▸ application-3 ▸ application.ts ▸ ...

```
1  import * as _ from 'lodash';
2
3  const result1 = _.head([1, 2, 3, 4]);
4  console.log(result1);
5
6  const result2 = _.head([]);
```

External NPM Modules

Demo Time! (2/4)

External NPM Modules

- Want to use NPM modules in our TypeScript projects
- Most NPM modules are just JavaScript, don't have type information
- We can create type definitions for each NPM module we use
  - Lots of duplicated work, across all TypeScript projects that use an NPM module

DefinitelyTyped

DefinitelyTyped / **DefinitelyTyped**

Watch 623 | Star 23,104 | Fork 18,116

Code | Issues 2,807 | Pull requests 110 | Projects 2 | Wiki | Security | Insights

The repository for high quality TypeScript type definitions. http://definitelytyped.org/

typescript | definition | dts | types | typings | typescript-definitions

63,407 commits | **54** branches | **0** releases | **9,029** contributors | View license

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find File** | **Clone or download**

Arik-neKrol and **armanio123** Type definitions for datatables.net-fixedcolumns (#36816) ··· Latest commit 6968096 29 minutes ago

.github	Update CODEOWNERS (#36703)	3 days ago
scripts	Add codeowners update (#33654)	4 months ago
types	Type definitions for datatables.net-fixedcolumns (#36816)	29 minutes ago
.editorconfig	Add Prettier, Lint Staged to help maintain repo consistency (#35672)	27 days ago

DefinitelyTyped

```
> npm install @types/lodash
```

DefinitelyTyped

Demo Time! (3/4)

Bundled Types

```
{
    "name": "awesome",
    "author": "Vandelay Industries",
    "version": "1.0.0",
    "main": "./lib/main.js",
    "types": "./lib/main.d.ts"
}
```

Bundled Types

Demo Time! (4/4)

Bundled Types

Write your NPM project in TypeScript?


TYPESCRIPT
LIGHTYEAR
TYPESCRIPT EVERYWHERE
imgflip.com

Summary

- TypeScript makes it easier to write JavaScript applications at scale
  - Introducing Static-Typing
    - Catch many common mistakes
  - Adding lots of tooling: auto complete, jump-to-def, refactoring etc…
- We need type definitions for the NPM packages we use, that are just written to JavaScript
- DefinitelyTyped is a massive repository of many type definitions for thousands of packages
  - All made available as @types/ packages on NPM
- More and more NPM packages are starting to bundle types
- More and more NPM packages are starting to be written in TypeScript
  - Removing the need for manually writing them, and ensuring they're **always in sync**

Rate the Session

How TypeScript is transforming the JavaScript ecosystem

Add to Your Schedule
Add Comment or Question

Sam Lanning (Semmle Inc)
11:00am-11:40am Thursday, July 18, 2019
Emerging Languages and Frameworks
Location: D135/136

Rate This Session

Who is this presentation for?

- Software developers and programmers (especially JavaScript programmers), software architects, CxOs and technical leads, and library and API designers

Level

Intermediate

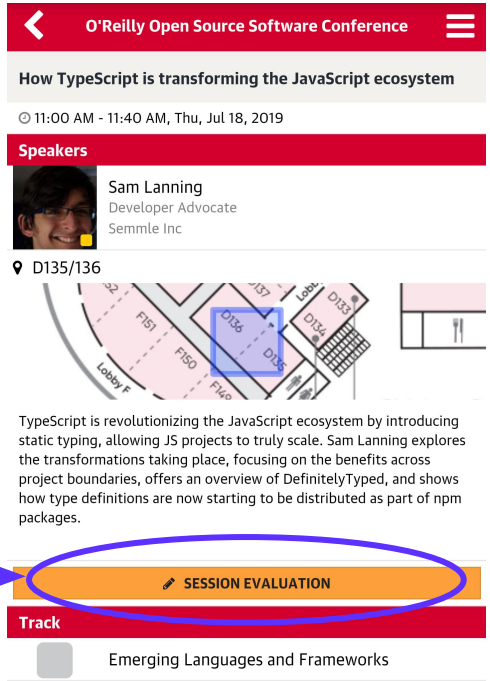
Description

TypeScript continues to grow in popularity—many projects have been migrating over to it from Vanilla JavaScript, and new projects are starting out as TypeScript projects from the get go. Its type system allows for many classes of mistakes to be found at compile time, and integrations with text editors and IDEs makes navigating large codebases a breeze.

But these beneficial TypeScript features can only be so useful if the dependencies and libraries that your project uses are not also considered; it should be possible to statically type-check your usage of the modules you import so features like code navigation and autocomplete work for these modules too.

The TypeScript community knows this, and there are a number of initiatives that ensure that type definitions are available for as many JavaScript libraries as possible. Sam Lanning explores TypeScript's motivations and goals. He then covers the two

Session page on conference website



O'Reilly Events App

Thank You

Writing npm (JavaScript) libraries using TypeScript

2:35pm–3:15pm Thursday

Live Coding ONLY

Portland 252

Follow along: github.com/samlanning/typescript-talks

- ➔ **Come get stickers!**
- ➔ **Rate the session**
- ➔ **Come to second talk**

Semmle™

http://**semmle.com**

@Semmle

Sam Lanning

@samlanning

@samlanning