TypeScript for UberConf 2018

@jessitron

Java + Scala + clojure (+ Ruby + bash + Elm + Elixir + Erlang + Fish + Kotlin)

+ TypeScript

-ATOMIST- ← software delivery automation at a higher level and in code

CEO: Rod Johnson, creator of Spring framework for Java

I'd rather you go away with

"that's off my list"

than

"wow that looks fun!

... wow this is painful"

↓ from the book How Information Grows

"Person-byte"

1. know if & when you should learn TS
2. be prepared for some WTFs
3. perspective on programming systems

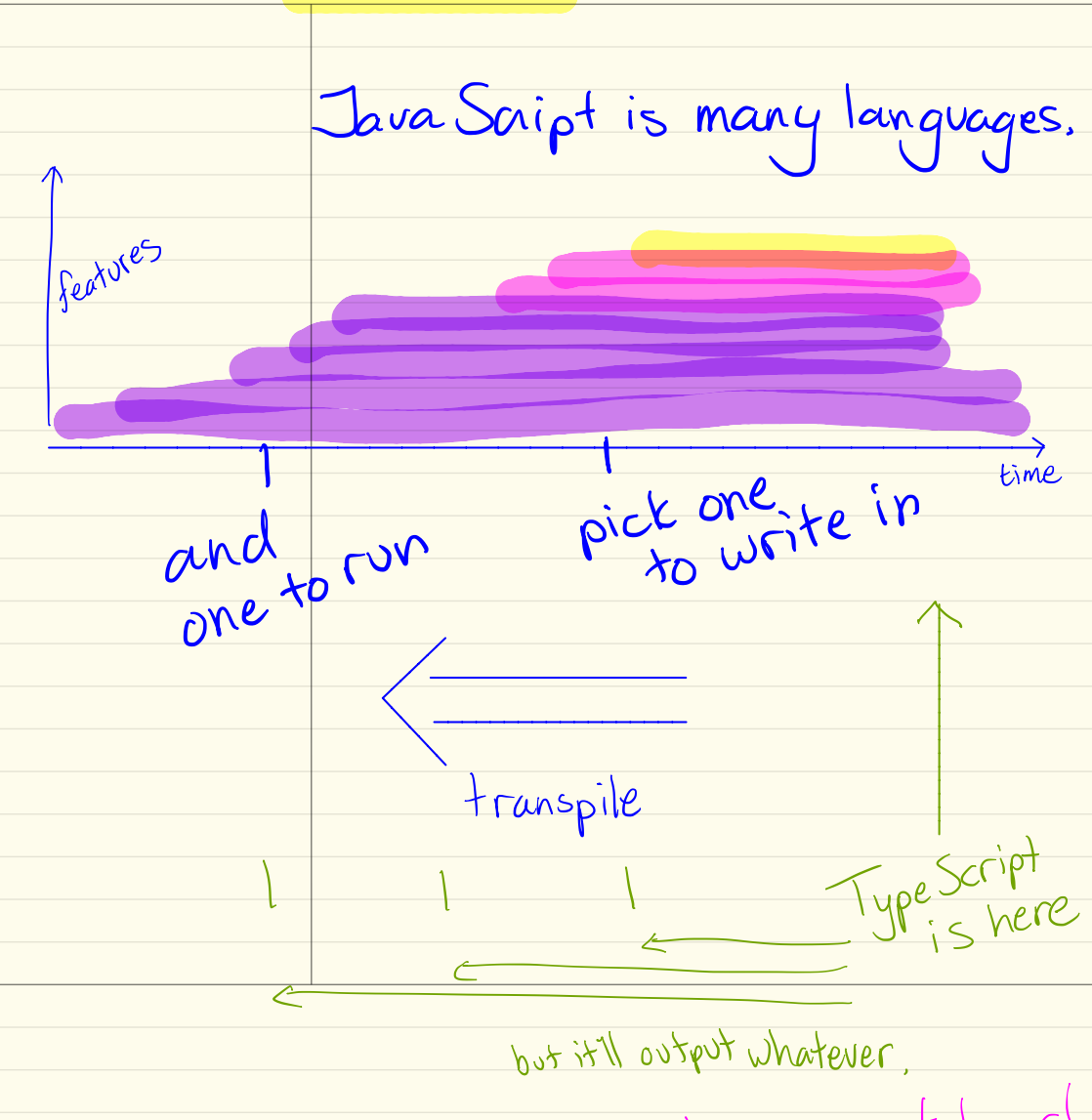
In scope:

types
node
npm
tslint
tests

Out of scope:

front end
http server

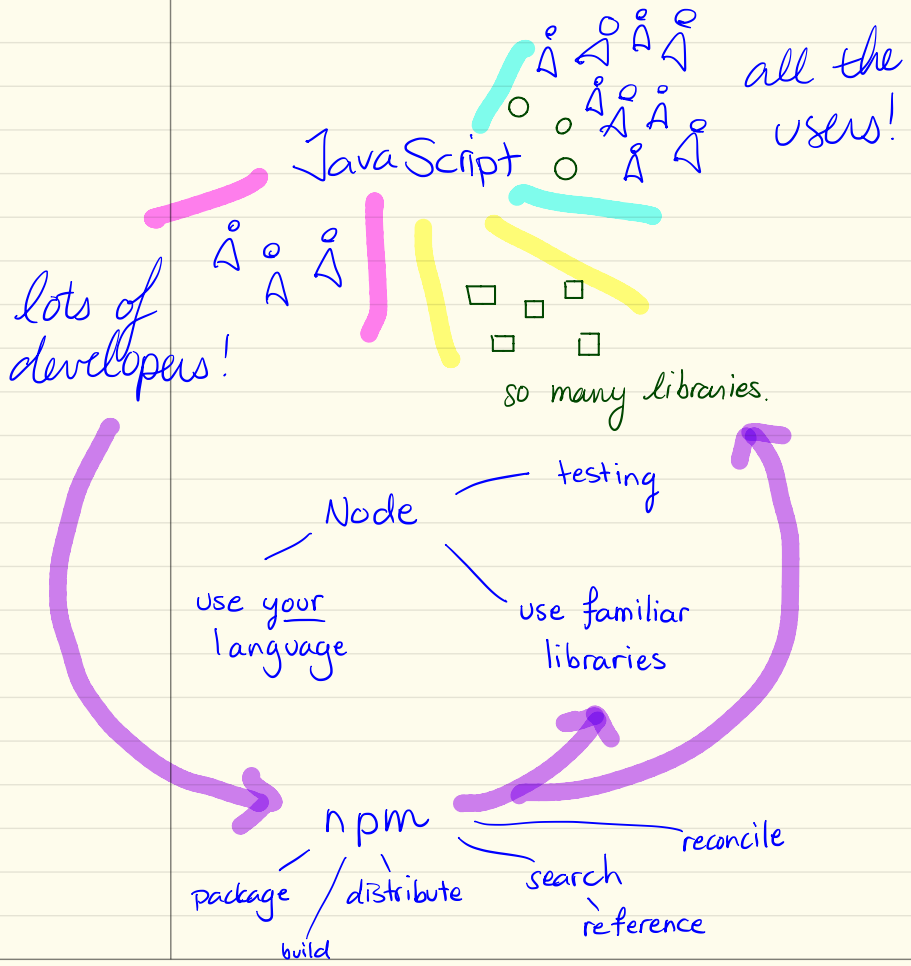
Useful



plus you get types!

but only when you want!

Useful!



We program in

Language Systems. and communities.

Live Coding Time

Explore One dungeon room.

1. synchronously
2. with callbacks
3. with Promises
4. and async/await
5. and "or" types
   and type guards.

next, Look at Code.

https://github.com/jessitron/explore-deps

start from explore.ts

click into `buildRoom`
and then
`tryToAct`

see notes in comments.

Where do types come from?

- @types / module
  special: @types/node
- typings files locally .d.ts
- those mysterious libs. in tsconfig
- /// reference
- make it output our type files!

↑ try removing the "*" from index.d.ts

Pause for syntax questions.

Where do modules come from?

TL;DR → it's a file that uses "import" or "export"

What is a "module"?

It's a unit of program composition.

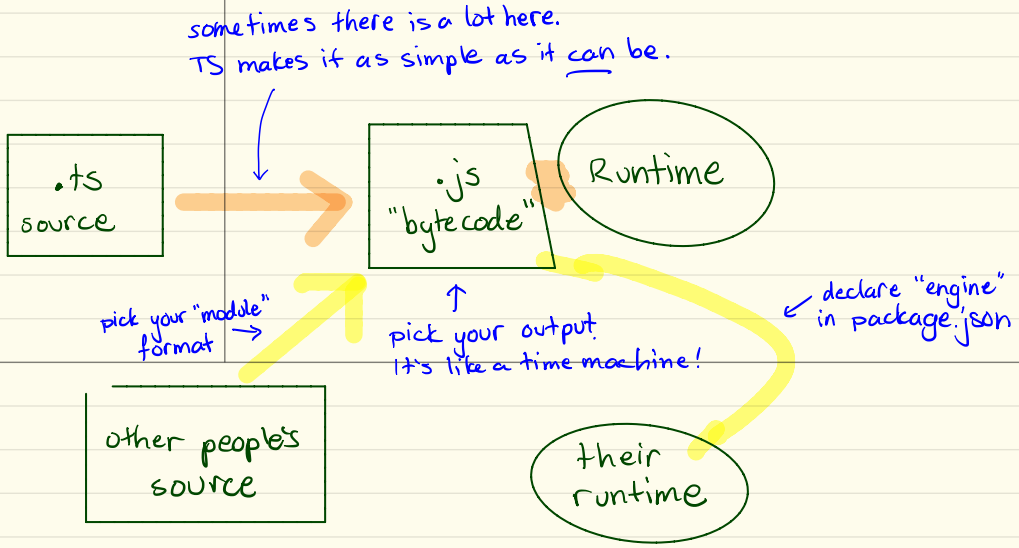
The level of re-use is the release.

JavaScript didn't come with modules.

See: amd, CommonJS, & tsconfig.module

↙ but node doesn't support it last I checked

ECMAScript now has a module standard



but, where do modules come from?

Ok, now we're talking runtime,
not compile-time.

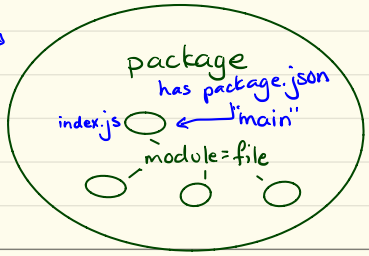
`require(...)`
↑ this is node, a function that is available globally within a module

`import ...`
↑ typescript syntax

Locating modules on disk:

1. relative to current file: `"./any/thing"`
2. package.
   now it gets exciting!

these live in npm and in your node_modules/ (plural.)



but how do they get in there?

npm install.

npm install

you can also use yarn.

we used to. but npm caught up.

dependencies
- runtime
- dev
- peer

tags
- npm view
  - --json specify fields

global is for command line utilities
- see "bin"

try `npm ls`

`npm link`

specifying a version:
- semver
- "*"
- ^ vs ~
- url

`npm ls` see a tree

`npm dedupe` reduce its size

npm scripts

`npm run x` test, build, etc

these use the executables in node_modules/.bin

cross-platform is hard
- npm-run-all instead of &&
- rimraf instead of rm

npm publish

set in package.json:

"bin"
or
"main" and "types"

"version"
↑
use `npm version patch` or 'minor' or 'major' to advance

"name": "@you/happy-project"

npm login

npm publish --access public

import statements

DO:

```
import * as fs from "fs";
```

DO:

```
import {readFile} from "fs";
```

then there are default exports:

lodash is friendly → you can also import * as _

```
import _ from "lodash";
```

boxen is a function. →

```
import boxen from "boxen";
```

one or more of these will work depending on the internals of the module you import.

Enforcing Correctness

Types:
start with `any`
and no null checking

narrow with compilerOptions

then! narrow further with static analysis

linting finds bugs.

you can also opt out of linting
for a line or file
for one rule or all

Testo:
how to mocha.

Useful

TypeScript is strictly more useful than JavaScript.

JavaScript is improving, but it'll always have warts. It has them for a reason: because people use it. And have been using it for decades.
It has warts because it's useful.

Compatibility matters.

We don't work alone.
Our teams don't work alone.
Our companies don't work alone.

We work in a network, of other coders and their past code
of users and their current runtime,
which is from the past.

TypeScript lets you work in the future while making it as easy as it can be to remain compatible with the past.

TypeScript

Live in the future.

Work with people of all eras.

Express yourself
and Check yourself with types
but only when they help you.

github.com/jessitron/explore-cleps