TypeScript

Rachael McQuater + Drew Colthorp

Atomic Object

```
git pull; npm install

Or

git clone https://github.com/atomicobject/strange-loop-ts-workshop.git
cd strange-loop-ts-workshop; npm install
```

Why TypeScript?

Why ~~TypeScript?~~ EcmaScript?

TypeError: undefined is not an object (evaluating 'undefined.foo')

— Some runtime exception

TypeScript is Modern EcmaScript

- Modules – `import`, `export`, etc.

- Async/await – Syntax for `promise.then(...)`

- Arrow functions – `x => x+1`

- Classes

- `for/of`

- Generator functions

- Decorators

- And more

TypeScript is Modern EcmaScript
with a Type System

```typescript
const x = 2
const y = "bar";
x * y // Error: y is not a number

interface Point {
    x: number,
    y: number
}
const point: Point = { x: 1 } // Error: y is missing on type Point
```

TypeScript is a compiler

- Like Babel: Compiles TypeScript to modern or older versions of EcmaScript

- You can even target IE 8

- It *also* compiles **EcmaScript** to EcmaScript. You can rely solely on TypeScript without Babel

TypeScript provides IDE infrastructure

Autocomplete, type checking as you type, and refactoring are built into the language server that's officially part of TypeScript.

Many editors provide some level of IDE-like support.

Visual Studio Code's is the best. The editor is written in TypeScript, so you'd hope so.

```typescript
1  const x = 2
2  const y = "bar";
3  x * y
4
5  interface Point {
6      x: number,
7      y: number
8  }
9  const point: Point = { x: 1 }
```

PROBLEMS 2   OUTPUT   DEBUG CONSOLE   •••   Filter. Eg: text, **

▲ TS Untitled-1 2
  ⊗ [ts] The right-hand side of an arithmetic operation must be of type 'any', 'num
  ⊗ [ts] Type '{ x: number; }' is not assignable to type 'Point'. Property 'y' is missi

TypeScript is configurable

TypeScript is configurable

- Targets many versions of EcmaScript

TypeScript is configurable

- Targets many versions of EcmaScript

- Configurable module behavior to interop with babel or node

TypeScript is configurable

- Targets many versions of EcmaScript

- Configurable module behavior to interop with babel or node

- Configurable strictness

TypeScript is configurable

- Targets many versions of EcmaScript

- Configurable module behavior to interop with babel or node

- Configurable strictness

- JSX Support

TypeScript is configurable

- Targets many versions of EcmaScript

- Configurable module behavior to interop with babel or node

- Configurable strictness

- JSX Support

- Can serve many purposes: find obvious errors in mostly-dynamic EcmaScript

TypeScript is configurable

- Targets many versions of EcmaScript

- Configurable module behavior to interop with babel or node

- Configurable strictness

- JSX Support

- Can serve many purposes: find obvious errors in mostly-dynamic EcmaScript

- Or strict static typing

Why TypeScript?

Why TypeScript?

- Analysis: navigate and understand foreign code.

Why TypeScript?

- Analysis: navigate and understand foreign code.

- Guides: make the correct path the easiest path while you're maximally immersed in the problem.'

Why TypeScript?

- Analysis: navigate and understand foreign code.

- Guides: make the correct path the easiest path while you're
  maximally immersed in the problem.'

- Safety: make most invalid states unrepresentable.

Why TypeScript?

- Analysis: navigate and understand foreign code.

- Guides: make the correct path the easiest path while you're maximally immersed in the problem.'

- Safety: make most invalid states unrepresentable.

- Rapid feedback: great editor support means all this happens *in your editor, as you type*.

```jsx
const bookMarkQuery = require("client/graphql/queries/BookmarkList.graphql");

const PureBookmarkList: React.SFC<PureProps> = props => {
  return (
    <div className="bookmark-list">
      <div className="bookmark-list-title">
        <T tag="header.recentlyUsedBookmars" />
      </div>
      {props.bookmarks.length > 0 ? (                    You, 6 months ago • Move bookmark butt
        <ul className="bookmark-list-items">
          {props.bookmarks.map(document => (
            <li key={document.id}>
              <a onClick={() => props.goToBookmarkedDocument(document.id)}>
                {document.title}
              </a>
            </li>
          ))}
        </ul>
      ) : (
        <div className="bookmark-list-none">
```

```typescript
type BlogId = Flavor<number, "BlogId">;
type PersonId = Flavor<number, "PersonId">;

function findBlog(id: BlogId) { /* ... */ }


const personId: PersonId = 1;
findBlog(personId);
```

Today