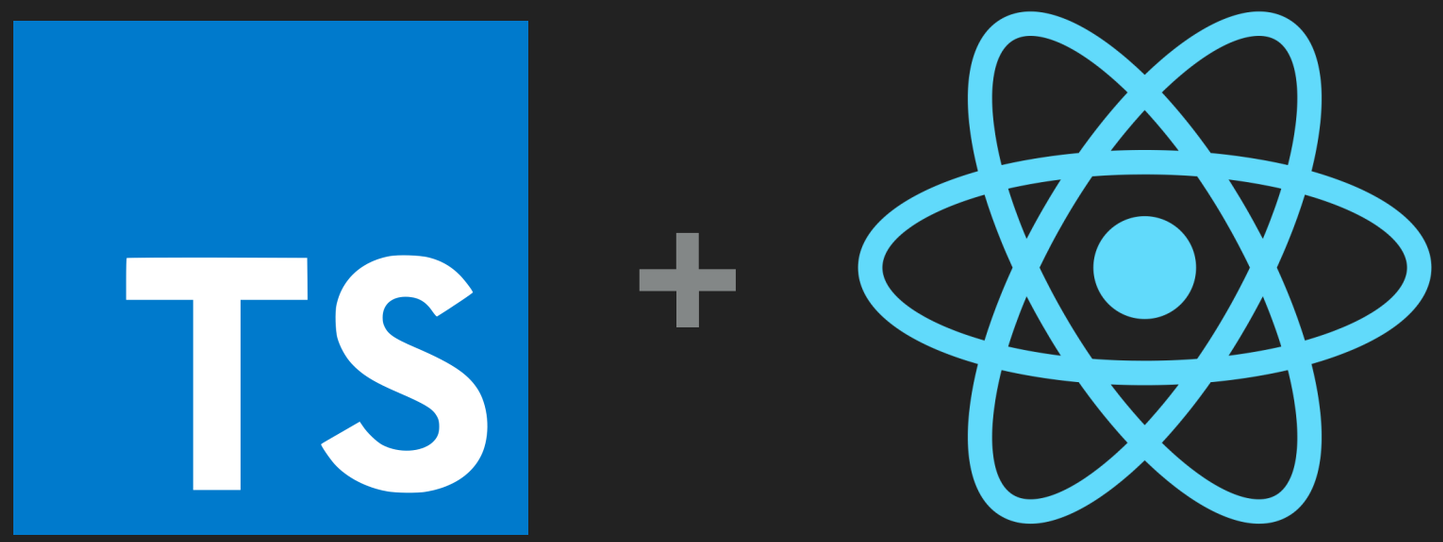


TYPESCRIPT IN REACT

HOW & WHY?

by Saulius Skeirys

ABOUT ME

- saulius.dev
- https://www.linkedin.com/in/sauliusskeirys/
- @saulske
- Software Engineer at NORTHERN TRUST

TS + 

WHAT IS TYPESCRIPT?

- Microsoft TypeScript is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript, and adds optional static typing to the language. TypeScript is designed for development of large applications and transcompiles to JavaScript. Wikipedia

TS + 

TYPESCRIPT IN SIMPLE TERMS

- Superset of JavaScript developed by Microsoft
- Optional typing language
- Compiles to plain Javascript
- Easily integrated into JavaScript projects
- Designed for development of large applications

TS + 

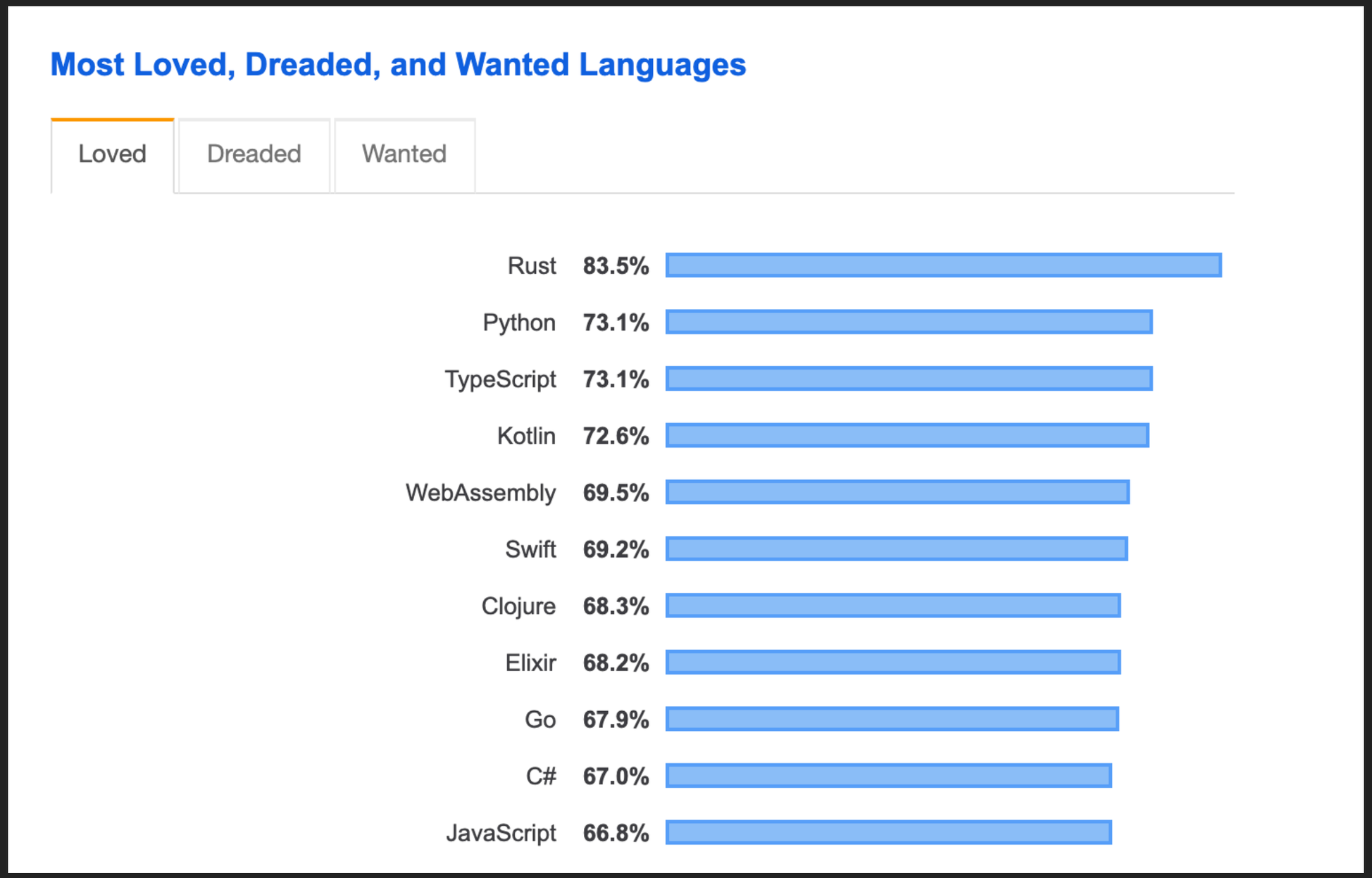


Image Source:
https://insights.stackoverflow.com/survey/2019/#most-loved-dreaded-and-wanted

TS + 

WHAT TYPESCRIPT OFFER?

- Strong Typing
- Object Oriented Features
- Compile-Time Errors
- ES6 Features
- Great Tooling

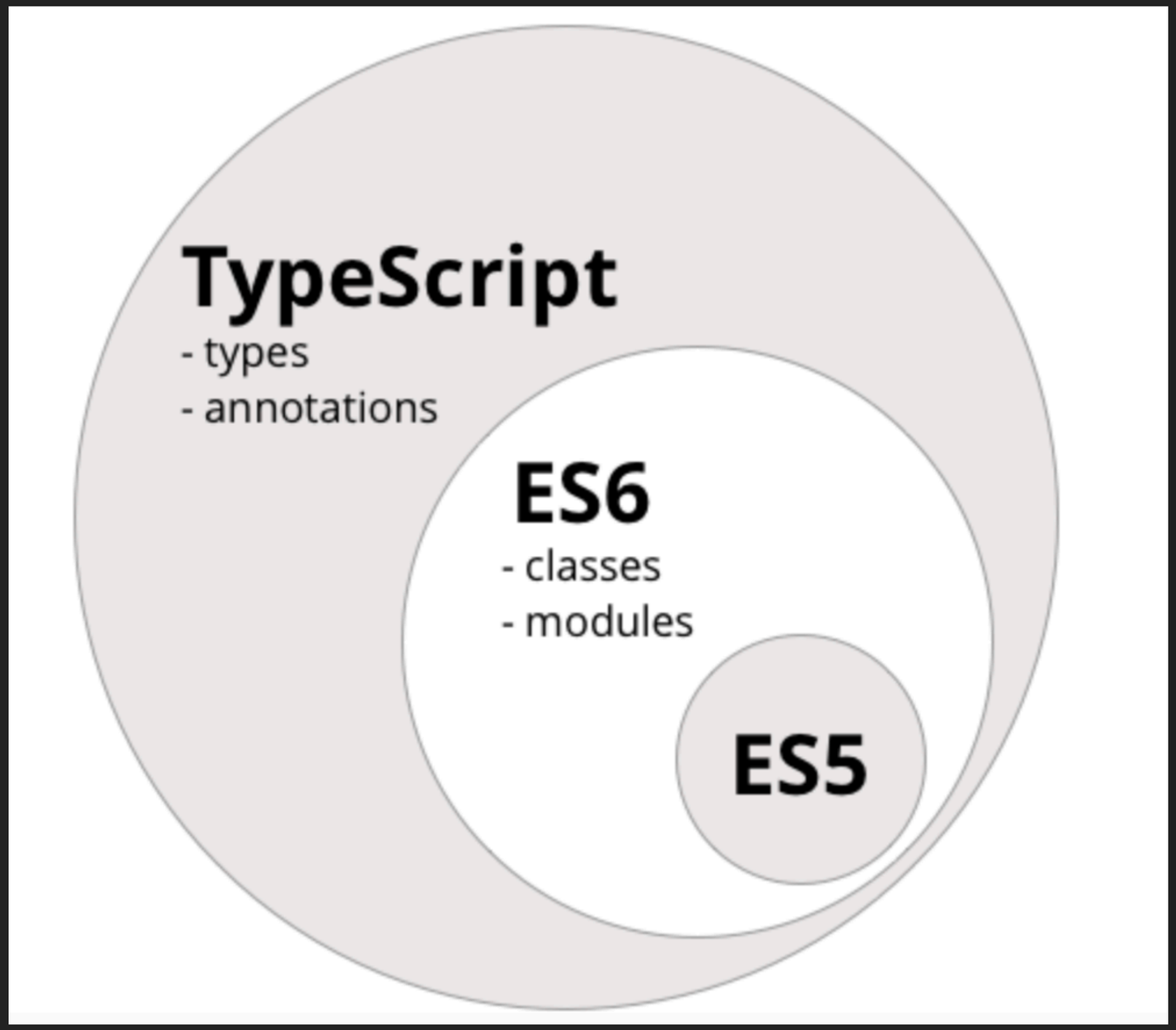


Image Source: https://www.ng-book.com/2/p/TypeScript/

TS + 

STATIC TYPING

```
1   // Car (Basic Typing)
2   const model: string = 'Jeep Wrangler';
3   const doors: number = 4;
4   const canOffroad: boolean = true;
5
```

STATIC TYPING

```
6   // Cars (Array Typing)
7   let cars: string[];
8   cars = ['Jeep Wrangler', 'Toyota FJ'];
9   cars.push(true);
10
```

TS + 

STATIC TYPING

```
11  // Can Car Offroad (Function Typing)
12  const canCarOffroad = (model: string) => {
13    switch(model) {
14      case 'Jeep Wrangler':
15        return true;
16      case 'Toyota FJ':
17        return false;
18      default:
19        return false;
20    }
21  }
22
23  canCarOffroad('Jeep Wrangler'); // return true
24  canCarOffroad('Toyota FJ'); // return false
25  canCarOffroad(undefined); // TS error
26
```

TS + 

STATIC TYPING

```
27  // Car (Interface)
28  interface Car {
29    model: string;
30    doors: number;
31    canOffroad: boolean;
32  }
33
34  const Car: Car = { model: 'Jeep Wrangler', doors: 4, canOffroad: true };
35  const FriendsCar: Car = { model: 'Toyota FJ', doors: false, canOffroad: false };
36
```

TS IN REACT

WHY?

- Catch problems early
- Code intellisense
- Refactoring features
- Code lookup
- Linter

CATCH PROBLEMS EARLY

Button.tsx ● App.tsx

src ▸ Button.tsx ▸ ⊸ ButtonProps ▸ 🔧 onClick

```
1   import React, {FunctionComponent} from 'react';
2
3   interface ButtonProps {
4     text: string;
5     disabled?: boolean;
6     onClick: () => void;
7   }
8
9   const Button: FunctionComponent<ButtonProps> = ({
10    text,
11    disabled = false,
12    onClick
13  }) => <button disabled={disabled} onClick={onClick}>{text}</button>;
14
15  export default Button;
```

TS + 

CATCH PROBLEMS EARLY

```
1  import React, {FunctionComponent} from 'react';
2  import Button from './Button';
3
4  import './App.css';
5
6  const App: FunctionComponent<{}> = () => {
7    return (
8      <div className="App">
9        <Button />
10     </div>
11   );
12 }
13
14 export default App;
15
```

TS + 

CATCH PROBLEMS EARLY

Button.tsx App.tsx

src ▸ App.tsx ▸ IFields ▸ firstname

```
1  import React, {FunctionComponent, useState} from 'react';
2  import Button from './Button';
3
4  import './App.css';
5
6  interface IFields {
7    firstname: string;
8  }
9
10 const App: FunctionComponent<{}> = () => {
11   const [fields, setFields] = useState<IFields>({
12     firstname: ''
13   });
14
15
```

TS + 

CODE INTELLISENSE

```
10
11  const App: FunctionComponent<{}> = () => {
12    const [fields, setFields] = useState<IFields>({
13      firstname: '',
14      lastname: ''
15    });
16
17    const handleFirstNameChange = (event: React.ChangeEvent<HTMLInputElement>) =>
18      setFields({ ...fields, firstname: event.target.value });
19
20    const handleSubmit = () => {
21      console.log('Fields', fields);
22    }
23
24    return (
25      <div className="App">
26        <input type="text" value={fields.firstname} onChange={handleFirstNameChange} />
27        <Button text="Submit" onClick={handleSubmit} />
28      </div>
29    );
30  }
31
32  export default App;
33
```

TS + 

REFACTORING FEATURES

Button.tsx × App.tsx

src ▸ Button.tsx ▸ ...

```
1   import React, {FunctionComponent} from 'react';
2
3   interface IButton {
4     text: string;
5     disabled?: boolean;
6     onClick: () => void;
7   }
8
9   const Button: FunctionComponent<IButton> = ({
10    text,
11    disabled = false,
12    onClick
13  }) => <button disabled={disabled} onClick={onClick}>{text}</button>;
14
15  export default Button;
16
```

TS + 

CODE LOOKUP

Button.tsx App.tsx

src ▸ App.tsx ▸ App

```
20    const handleLastNameChange = (event: React.ChangeEvent<HTMLInputElement>) =>
21      setFields({ ...fields, lastname: event.target.value });
22
23    const handleSubmit = () => {
24      console.log('Fields', fields);
25    }
26
27    return (
28      <div className="App">
29        <input type="text" value={fields.firstname} onChange={handleFirstNameChange} />
30        <input type="text" value={fields.lastname} onChange={handleLastNameChange} />
31        <SubmitButton text="Submit" onClick={handleSubmit} />
32      </div>
33    );
34  }
35
36  export default App;
37
```

TS + React

BENEFITS OF USING TS IN REACT

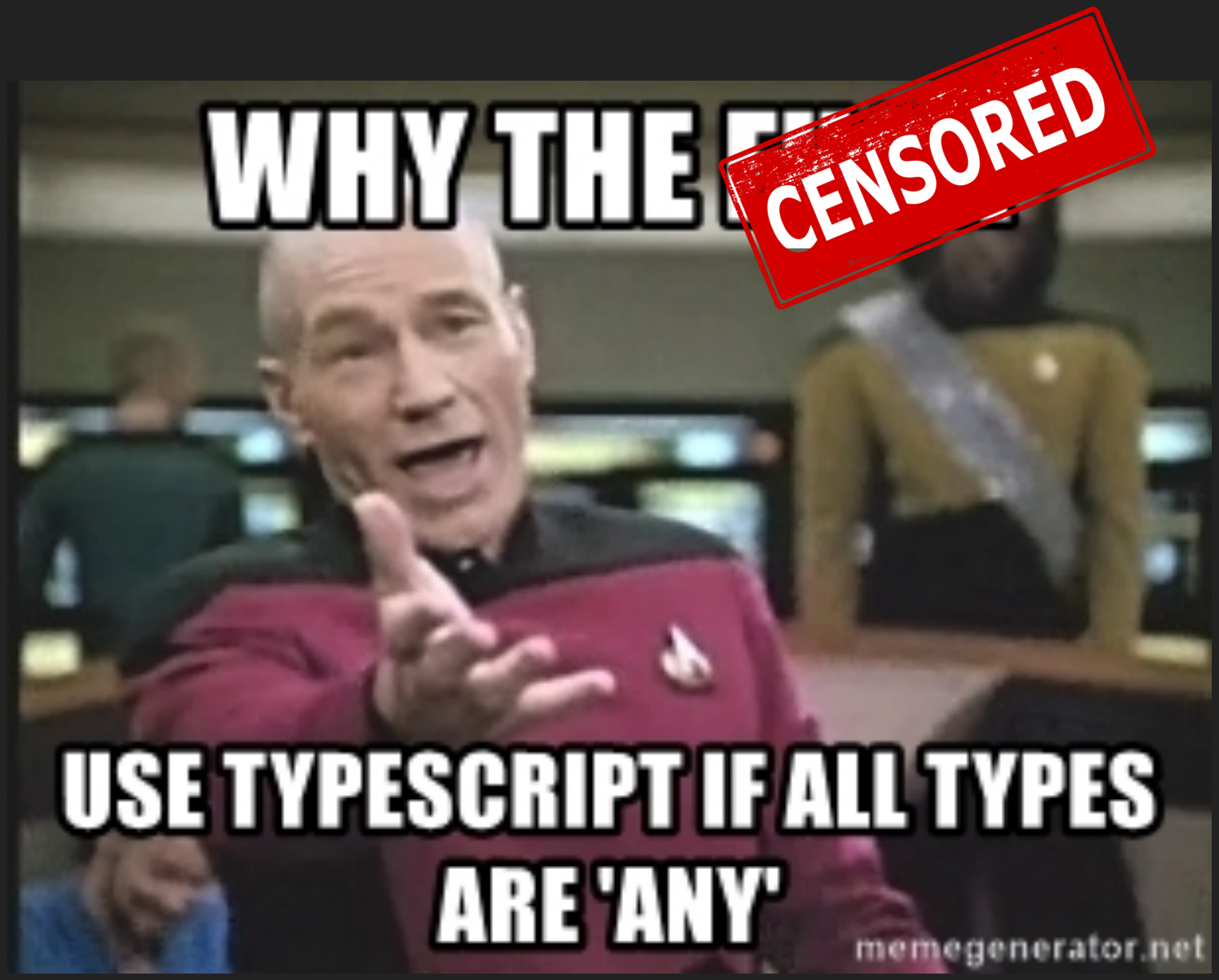
- Readability and validation
- Interfaces
- Refactoring
- Less bugs
- Pushes developer to have correct workflow
- ECMAScript 2015 and future proposals support

CONCERNS OF USING TS IN REACT

- Recruitment*
- On-boarding time*
- A bit difficult to setup
- Extra code
- TS vs. Flow?

WHO SHOULD USE IT?

- Companies with sizeable teams
- Large Codebase
- Who loves typing


WHY THE
CENSORED
USE TYPESCRIPT IF ALL TYPES
ARE 'ANY'
memegenerator.net

TS + 

DEMO: https://github.com/saulske/ts-react-demo

THANK YOU