Dawn of REASON

@sander_spies

The earliest prototypes of

React

were written in

Standard ML

2013

```
var Foo = React.createClass({
  render: function() {
     return <Bar />;
  }
});
```

Our quest for maintainable applications has

led us to similar concepts found in ML

```
let peek = cat => {
  if (Math.random() > 0.5) {
    cat.lives--;
  }
};
```

```
/* @flow */
type schrodingersCat = {
 lives: number
};

let peek = (cat:schrodingersCat) => {
  if (Math.random() > 0.5) {
    return {...cat, lives: cat.lives - 1};
  }
  return cat;
};
```

```
/* @flow */
type animal = "cat" | "dog";

let petAnimal = (animal:animal) => {
  switch (animal) {
    case "cat":
      /*...*/
      break;
    case "dog":
      /*...*/
      break;
  }
};
```

```
<Match pattern="/:user" render={(matchProps) => (
  <div>
    <Match pattern="/about" component={About}/>
    <Match pattern="/company" component={Company}/>
 </div>
)}/>
```

Types, immutability and pattern matching

reduce accidental complexity

BABEL

2016

+

BABEL

flow

REACT ROUTER

Meanwhile at…

Concurrent React Prototype in OCaml

@jordwalke

OCaml

Functions

Types

Immutable by default

Pattern matching

Compiler toolchain

Catch issues at compile time

Compile to JS/native/kernel

Also objects, classes, modules, language extensions, and more has the defaults we want

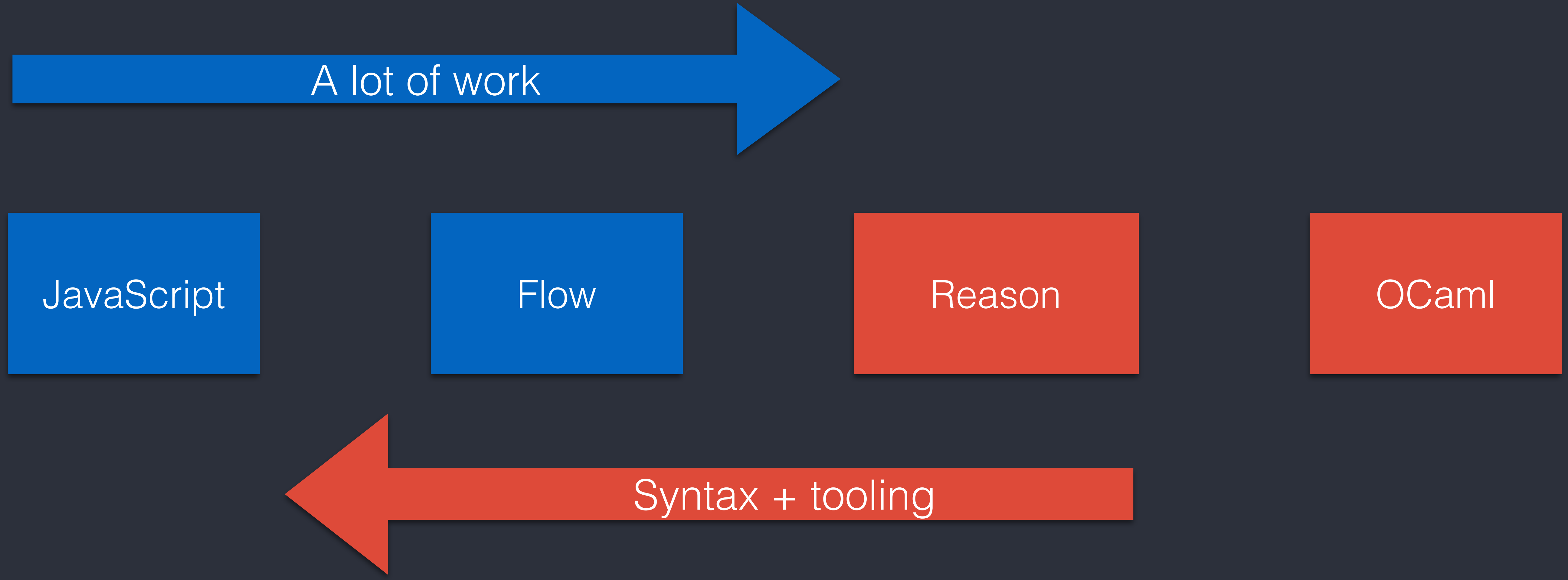
JS developer trying to grasp OCaml syntax

What if…

REASON

BUILD SYSTEMS RAPIDLY

Adopting features of ML



Becoming familiar to JS developers

“Let’s drop everything I know”

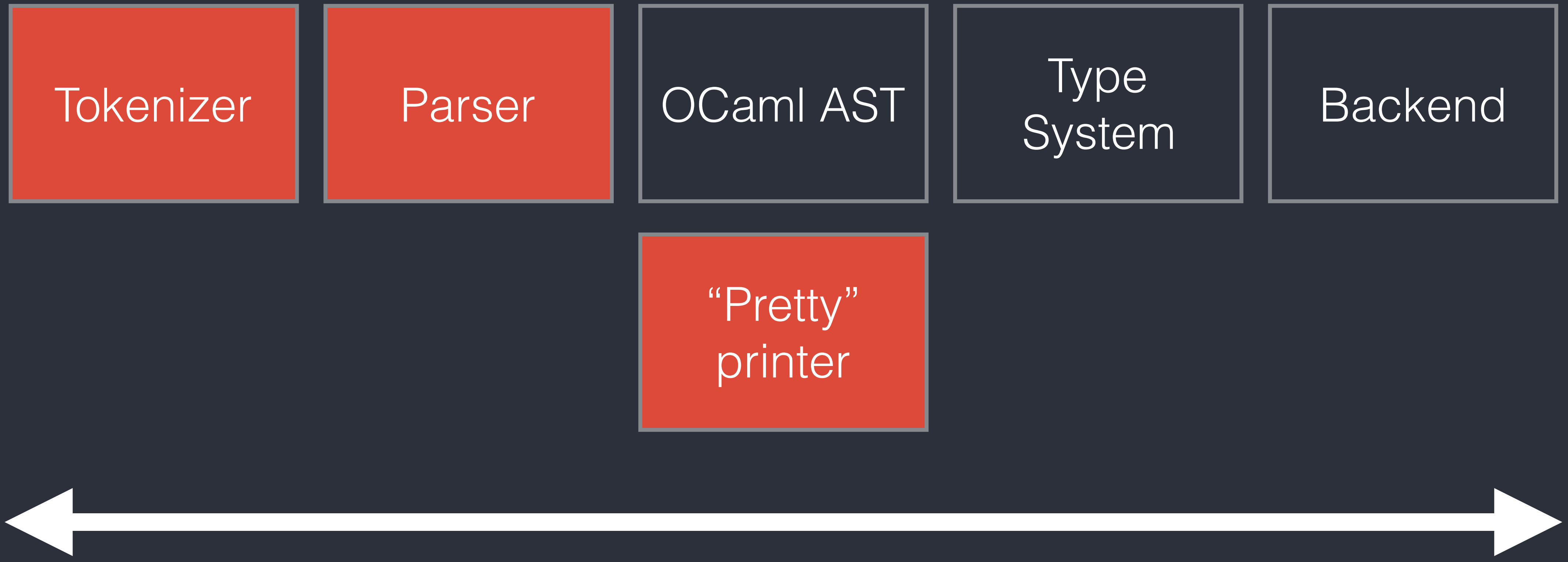
- nobody ever

Syntax

Build tooling

Sharing

Syntax



OCaml compiler toolchain

Rebel

git clone https://github.com/reasonml/RebelExampleProject

Use package.json to configure everything

Target web or native

Editor support

Vim, Emacs, Atom, Sublime and soon VS Code

Building on shoulders of existing OCaml tools

Other tooling

rtop - a repl for Reason

refmt - pretty printer

rejs - JS to Reason

Example

```
type schrodingersCat = {
 lives: int
};

let peek cat => {
  if (Random.bool ()) {
   {...cat, lives: cat.lives - 1};
  }
  else {
   cat;
  }
};
```

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

let petAnimal animal => {
  switch (animal) {
    | Cat => {}
    | Dog => {}
  }
};
```

Building on familiarity

Reason with React Bindings
Preview

Get started:

https://github.com/reasonml/RebelExampleProject

Ask questions:

https://gitter.im/facebook/reason

@sander_spies