React ❤ TypeScript

N.Pirotte - Front-end Craftsman @ Wemanity

github.com/**npirotte**

/typescript-webpack-react-flux-todo

What we will speak about

1. **Tooling**
    1. Packaging
    2. External libraries usage (TSD)
2. **Build a React app with TypeScript**
    1. Components creation
    2. In-deep review of the Component typing mechanism
    3. React Syntetic Events typing

Starting point of TS + React experience :

Get the right tools !

How to build a TS/React app ?

Use **Webpack** with a **TypeScript loader**

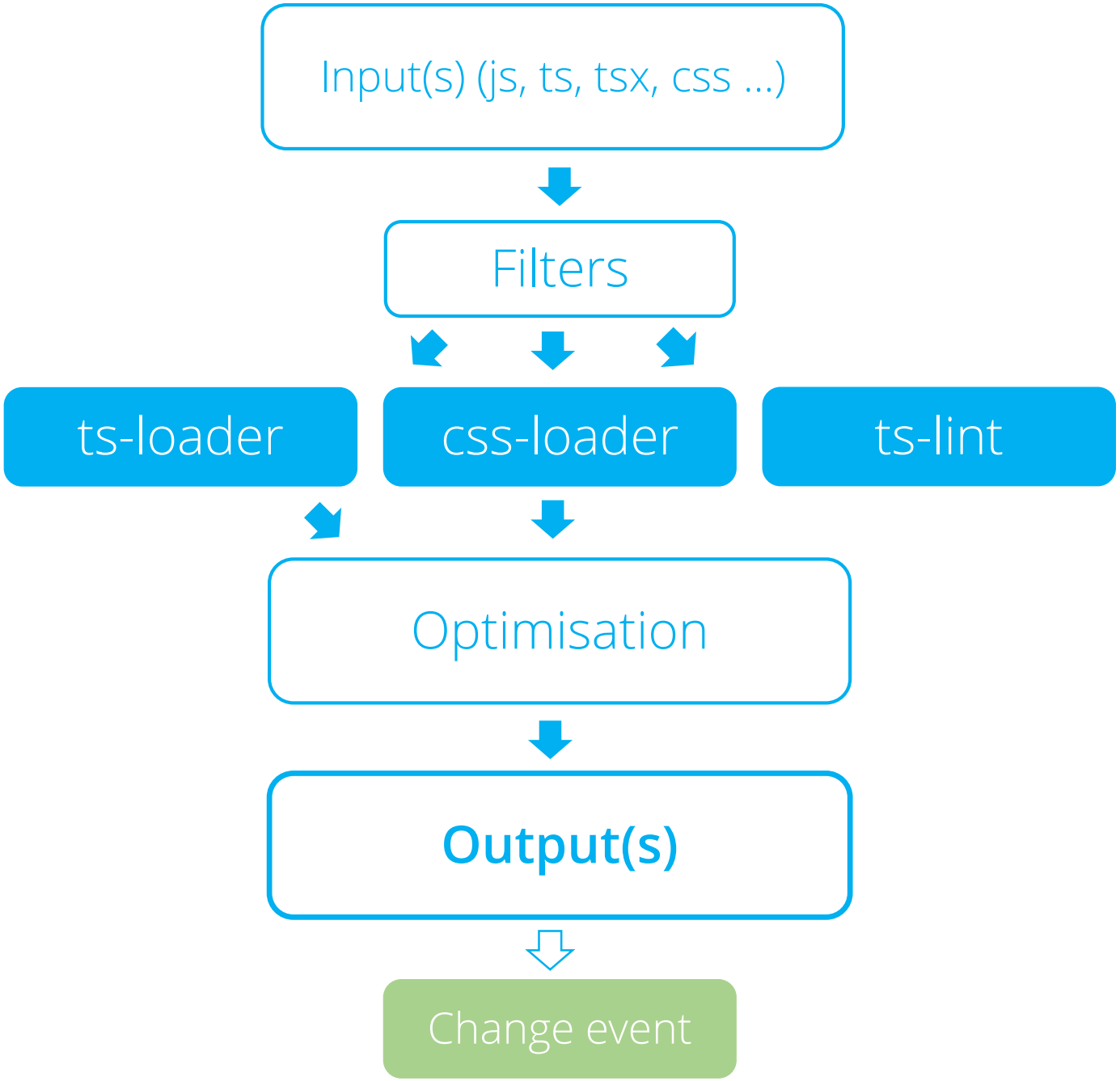
- TS to JS ES5 conversion
- JSX (TSX) to JS conversion (yes, Microsoft did it !)
- Production ready compilation
- And all awesomness of Webpack
  - Hot reload
  - Pre/Post loaders (lint, optimisation, stylesheets loader etc.)

Pick an example on my github for the webpack.conf.js !

How to build a TS/React app ?

Inside webpack.conf.js :

```
loaders: [
    {test: /\.ts(x?)$/, loader: 'awesome-typescript-loader?instance=jsx’},
    {test: /\.scss/, … }
]
```


Input(s) (js, ts, tsx, css ...)
Filters
ts-loader
css-loader
ts-lint
Optimisation
Output(s)
Change event

How to get **typing** of **React** and Other Libraries ?

Typing of third-party libraries

Use the awesome **Definitly Typed** plugin (tsd)

- Nice CLI tool
- Thousands of libraries available (including React, Flux etc)
- Easy type dependencies management

Ok, let’s write some **great code** !

A simple **TODO App**

- **App**
  - ContentPage
    - TaskFormComponent
    - TaskListComponent
      - TaskComponent
      - TaskComponent
      - ....
    - TaskSummaryComponent

A simple **TODO App**

App

- ContentPage
  - TaskFormComponent
  - TaskListComponent
    - TaskComponent
      - **TaskComponent**
      - ....
  - TaskSummaryComponent

??? React.Component<{}, {}> ???

TypeScript **generic** types

- Dynamic interface declaration
- Allow reusable interfaces !
- Similar to java and C#

```
interface Array<T> {
    [n: number]: T;
    concat(...items: T[]): T[];
    sort(compareFn?: (a: T, b: T) => number): T[];
    …
}
```

Synthetic events with TypeScript

React events interfaces

- ClipboardEvent
- CompositionEvent
- DragEvent
- FocusEvent
- FormEvent
- KeyboardEvent
- MouseEvent
- TouchEvent
- UIEvent
- WheelEvent

From React to **native DOM**

- Use TypeScript **as** keywork
- Apply native **HTML{ElementName}Element** type

```
const elm: HTMLInputElement = evt.target as HTMLInputElement;
```

github.com/**npirotte**