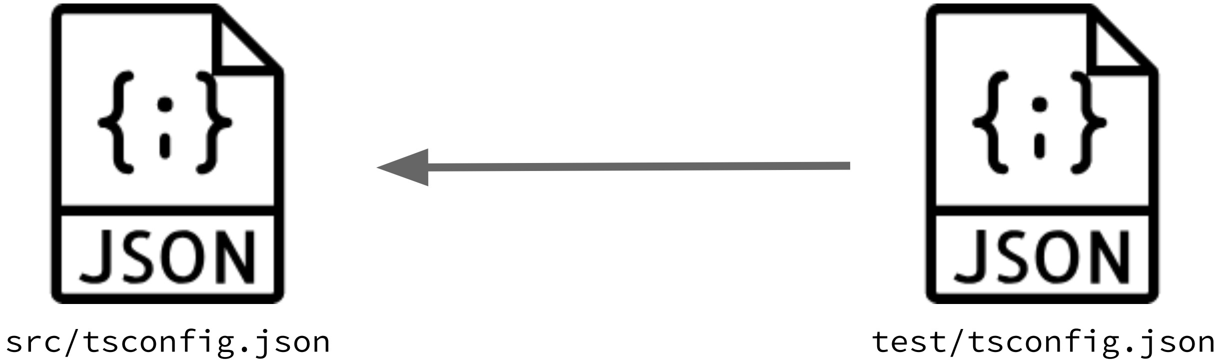
TypeScript 3.0

What's New?

Project References

Link projects to each other



- Faster build times
- Easier builds
- More logical separation of code

Project References

‘unknown’ type

Example:

```
const mysteryData: unknown = getMysteryData();
const val0 = 5 + mysteryData;
if (typeof mysteryData === 'number') {
  const val1 = 6 + mysteryData;
  console.log(val1);
}
```

‘unknown’ type

LibraryManagedAttributes

Official definition:

- TypeScript 3.0 adds supports a new type alias in the `JSX` namespace called `LibraryManagedAttributes`. This helper type defines a transformation on the component's `Props` type, before using to check a JSX expression targeting it; thus allowing customization like: how conflicts between provided props and inferred props are handled, how inferences are mapped, how optionality is handled, and how inferences from differing places should be combined

And usage in @types/react:

```
type LibraryManagedAttributes<C, P> = C extends { propTypes: infer T; defaultProps: infer D; }
    ? Defaultize<MergePropTypes<P, PropTypes.InferProps<T>>, D>
    : C extends { propTypes: infer T; }
        ? MergePropTypes<P, PropTypes.InferProps<T>>
        : C extends { defaultProps: infer D; }
            ? Defaultize<P, D>
            : P;
```

Before

```tsx
export interface Props { count?: number; }
export class Counter extends React.Component<Props> {
  static defaultProps = { count: 0 }
  render() { return <div>Count: {this.props.count! + 1}</div>; }
}
let counter = <Counter />;
```

After

```tsx
export interface Props { count: number; }
export class Counter extends React.Component<Props> {
  static defaultProps = { count: 0 }
  render() { return <div>Count: {this.props.count + 1}</div>; }
}
let counter = <Counter />;
```

LibraryManagedAttributes

Tuple type improvements

Before

```
const tuple: [number, string] = [1, "str"];
const func = (num: number, str: string) => num + str;
func(...tuple);
```

After

```
const tuple: [number, string] = [1, "str"];
const func = (num: number, str: string) => num + str;
func(...tuple);
```

Also cool things like:

```
declare function bind<T, U extends any[], V>(
  f: (x: T, ...args: U) => V,
  x: T,
): (...args: U) => V;
declare function f3(x: number, y: string, z: boolean): void;
const f2 = bind(f3, 42);  // (y: string, z: boolean) => void
const f1 = bind(f2, "hello");  // (z: boolean) => void
const f0 = bind(f1, true);  // () => void
```

Tuple type improvements

TypeScript 3.1

What’s New?

Properties on functions

Before

```
const NumRenderer = (props: { num: number }) =>
  <span>{props.num}</span>;
NumRenderer.defaultProps = { num: 5 };
```

After

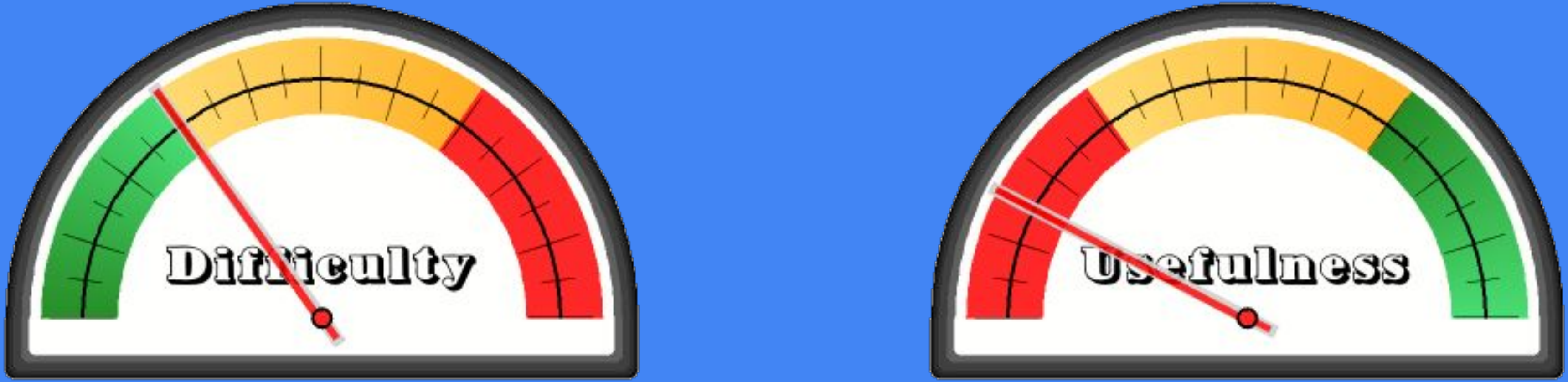
```
const NumRenderer = (props: { num: number }) =>
  <span>{props.num}</span>;
NumRenderer.defaultProps = { num: 5 };
```

Properties on functions

typesVersions

Select code based on TypeScript version:

```json
{
  "name": "package-name",
  "version": "1.0",
  "types": "./index.d.ts",
  "typesVersions": {
    ">=3.1": { "*": ["ts3.1/*"] },
    ">=3.5": { "*": ["ts3.5/*"] }
  }
}
```

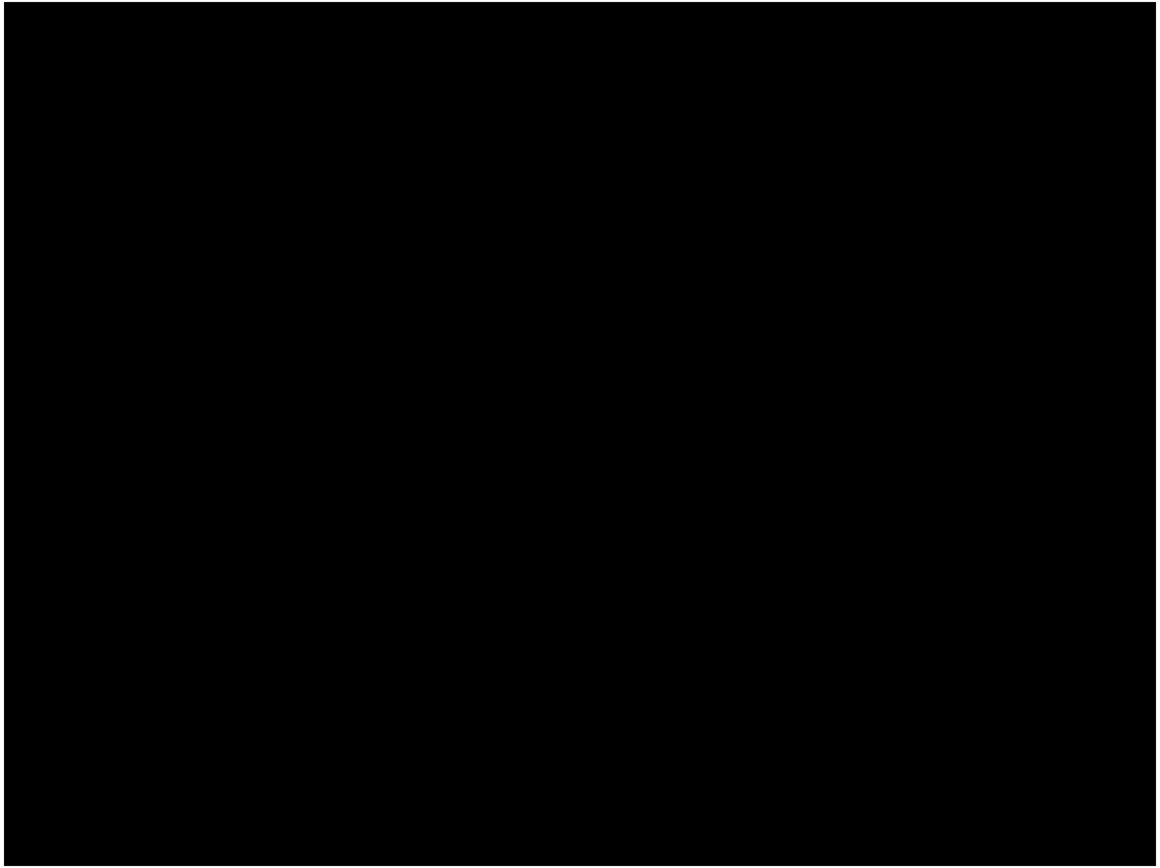


typesVersions

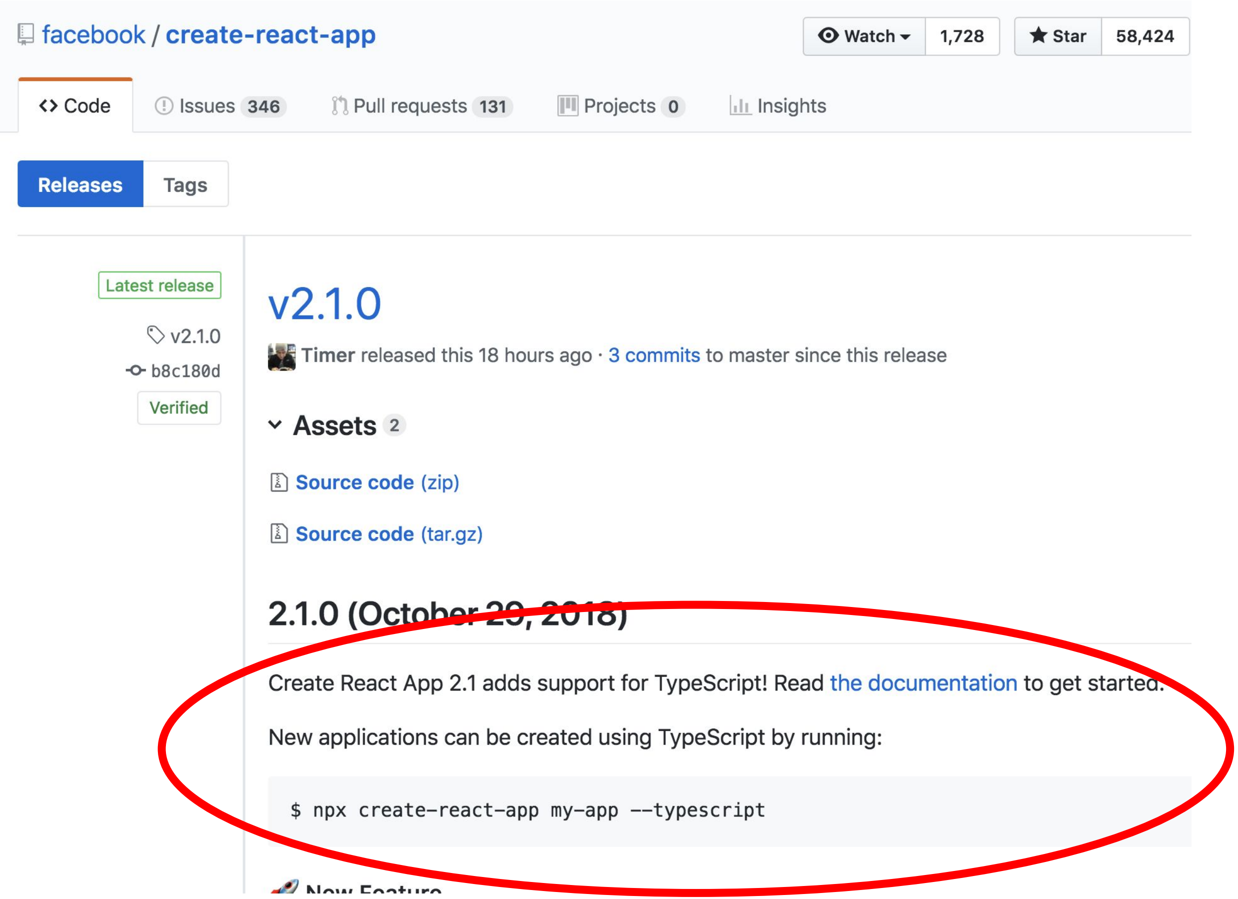
Bonuses!

What's New?

Async refactoring



create-react-app TypeScript support



Thanks!

Gauge image credit to
http://www.optixl.com/gauge/index.php

Learn more at
https://github.com/Microsoft/TypeScript/wiki/What's-new-in-TypeScript#typescript-31

Created by
Jason Killian
jasonkillian.com