**Clarifying Which Babel to Use for Compiling React Today**

[Babel](https://babeljs.io/docs/en/) is a toolchain to transpile new JavaScript syntax (e.g. ES6) down to older versions that is supported by browser or node.js execution environments. The latest major version at the moment is [Babel 7](https://babeljs.io/docs/en/v7-migration). To transpile JavaScript React, we normally use bable.

The problem is that babel is evolving so fast that it can be confusing to know what to use. Especially when you are supporting older React application as well as creating new ones. Is stage-x still a thing? What happens to preset-es2017? Why my babel has @babel prefix? What should I use?

If you are starting a new React project, my recommendation is to use the latest set. I have been guilty of using create react app or get other developers to configure the environment and was pretty confused about it. It took me a while to figure out what had been deprecated and what would be the best to use in my own React project. After burning my fingers a few times, I have a little bit better knowledge about this toolchain.

Before going any further, I have to note that this is for compiling React with the latest webpack 4 and babel 7 to date. In this space, things move very quickly and everything becomes legacy in a few months time.

Now, it’s time to demystify some of the things that confused me as well as the tips so that you don’t waste your time.

1. **stage-x and esYear modules are deprecated.**

We do not need to use stage-x or esYear modules. They are replaced by @babel/preset-env module.

1. **@babel prefixed modules are the latest ones.**

If you are using something like babel-preset-env or babel-preset-stage-2, they are super old and not compatible with webpack 4. You will see them in legacy React apps with older webpack versions where developers are too scared to upgrade. You can spot them when you look at presets in babelrc files. They are short like [“preset-env”, “stage-2”]. The latest ones has prefix as [“@babel/preset-env”].

1. **preset-env rules**

Instead of yearly presets like es2017 or stage-x presets, use @babel/preset-env. It includes all the deprecated presets.

1. **Don’t worry about the target for preset-env if you don’t have specific requirements**

With preset-env, you can specify the target as in the documentation [here](https://babeljs.io/docs/en/babel-preset-env). Note that the default is the browserlist. So, if you want to target browsers, you do not need to specify the target.

Basically, this one is the same

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

"presets": [

[

"@babel/preset-env",

{

"targets": "> 0.25%, not dead"

}

]

[/cc]

As this one

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

"presets": ["@babel/preset-env"]

[/cc]

1. **Update babel often**

If you are re-using package.json from a few month ago, it is probably outdated. You can upgrade babel with npx babel-upgrade

1. **@babel/plugin-proposal-class-properties are needed for using arrow function in React class**

Using arrow functions in React class is definitely preferable. Lexically scoped class functions are easier to use and make code more concise. Unfortunately, this is not supported out of the box preset-env. You need to install the plugins.

1. **Read the error message**

When compilation fails, babel gives you pretty good error message. Before jumping to google, make sure to read them. The answer is right in front of your eye.

1. **Create .babelrc file instead of putting in package.json file**

It is recommended by babel. The separation makes it maintainable because package.json file can get pretty big.

**Setup Example**

Here is the example of babel set up. This is the minimum requirement to write modern React application.

**.babelrc**

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

**Versions**

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

Now we can finally stop worrying about Babels and start building cool stuff … until the whole thing becomes obsolete in a few month time… urrg…