Babel: A JavaScript Compiler for Modern Development

Babel is a powerful JavaScript compiler that serves a crucial role in modern backend development. Its primary function is to transform code written in newer ECMAScript versions (ES6, ES7, etc.) into a format that's compatible with older JavaScript environments, particularly Node.js versions that might not natively support the latest features.

❖ Babel's Processing Steps in Backend JavaScript Files

1. **Parsing:** Babel starts by examining the input JavaScript code and creating an internal representation (often an Abstract Syntax Tree or AST) that captures the code's structure and meaning.

2. **Transformation:** Based on your configuration, Babel applies various plugins and presets to modify the AST. This transformation can involve:
    - ***Transpiling:*** Converting newer syntax elements (like arrow functions, const and let declarations, class syntax) into their equivalent ES5 counterparts that older Node.js versions can understand.

    - ***Polyfilling:*** Adding code snippets (polyfills) to compensate for missing features in the target environment. For instance, if your code uses the fetch API, which isn't universally supported, Babel might inject a polyfill to ensure that fetch functions as expected.

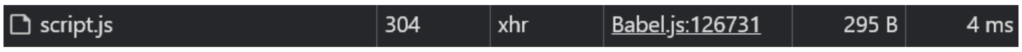
    - ***Minification (Optional):*** Babel can optionally be configured to minify the code, meaning it removes whitespace and unnecessary characters to reduce file size and improve performance.

❖ Using Babel

We can use Babel by including cdn link into our html script tag or copy-paste link context into **Babel.js** file.
And make changes in script tag of **script.js** file as shown:

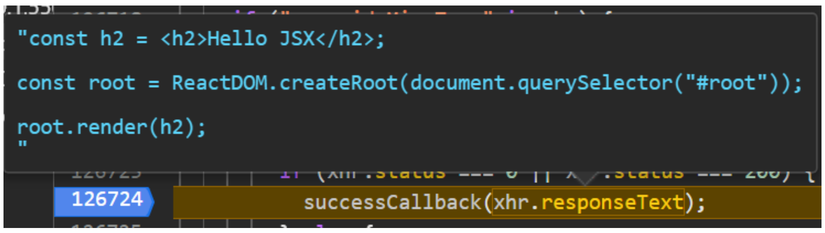
```html
<script src="./script.js" type="text/babel"></script>
```

After we run using this the babel adds extra script tag on execution which have content in the form of "React.createElement" (check it on inspect elements).\



```
script.js    304    xhr    Babel.js:126731    295 B    4 ms
```

When we used debugger on the babel.js we get to know it's taking our whole script.js file as string, like it's showing:

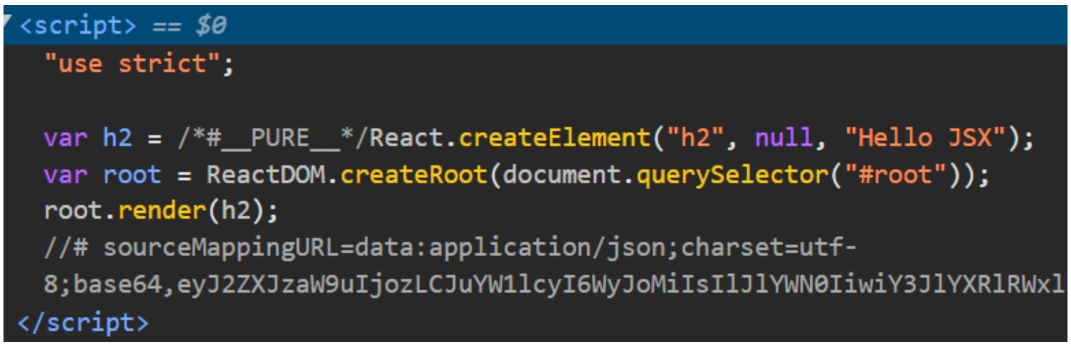


```
"const h2 = <h2>Hello JSX</h2>;

const root = ReactDOM.createRoot(document.querySelector("#root"));

root.render(h2);
"
126724        successCallback(xhr.responseText);
```

Now after taking our whole script.js file Babel converts it to react.js file or compile the file as we can see it's returning script tag to html, where our compiled code is written:



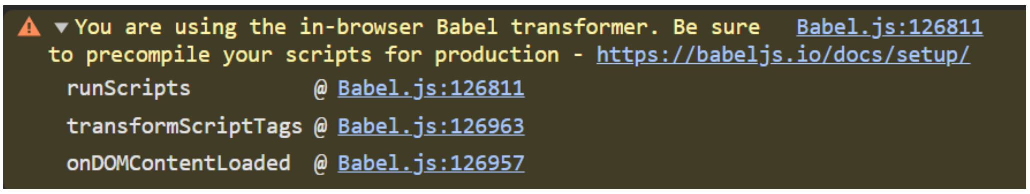
```
<script> == $0
  "use strict";

  var h2 = /*#__PURE__*/React.createElement("h2", null, "Hello JSX");
  var root = ReactDOM.createRoot(document.querySelector("#root"));
  root.render(h2);
  //# sourceMappingURL=data:application/json;charset=utf-
  8;base64,eyJ2ZXJzaW9uIjozLCJuYW1lcyI6WyJoMiIsIlJlYWN0IiwiY3JlYXRlRWxl
</script>
```

It also have **Source Mapping URL,** we'll understand about it later on.

As we already know our index.html file don't process our original written **"script.js"** file as we know that while creating it's script tag we've set it's **type = "text/babel",** that's why html not processing it because **HTML only process file of type of Javascript i.e. type = "text/javascript".**

So that's the advantage of using Babel, with the help of it we can write our HTML code in JS file,
But this method is not recommended, the more authentic method is to use babel through NPM (Node Package Manager).



```
⚠ ▼You are using the in-browser Babel transformer. Be sure   Babel.js:126811
   to precompile your scripts for production - https://babeljs.io/docs/setup/
    runScripts          @ Babel.js:126811
    transformScriptTags @ Babel.js:126963
    onDOMContentLoaded  @ Babel.js:126957
```

❖ Babel Setup:

1. **npm init –y**

2. **npm install --save @babel/standalone**

3. **npm install -g babel-cli**

4. **npm install --save-dev @babel/core**

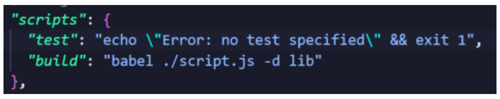
5. **Create a ".babelrc" file and paste following command in it.**

```
{
 "presets": ["@babel/preset-env", "@babel/preset-react"]
}
```

6. **npm i @babel/preset-env -D**

7. **npm i @babel/preset-react -D**

8. **npm install --save-dev @babel/core @babel/cli @babel/preset-env @babel/preset-react**

9. **Make changes in package.json file**  **like this in build.**

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "build": "babel ./script.js -d lib"
},
```

10. **npm run build**