ES6

- ES6 IS ES2015(MORE ADVANCED VERSION OF JAVASCRIPT.)
- **ES6 to ES5 for browsers to understand**
  - Transpilers (BABEL, Traceur...)

- **Installation**
  - Node.js and Visual Studio Code
  - \`npm –version\` to check whether latest version is installed.
  - \`npm init\` in terminal to initialize package.json in the working directory. After package.json is created add all the dependency package names and their version is to be added in the .json file.

```
{
  "name": "hello_app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "radhika",
  "license": "ISC",
  "devDependencies":{
    "webpack":"1.14.0",
    "babel-core":"6.21.0",
    "babel-loader":"6.2.10",
    "babel-preset-es2015":"6.18.0",
    "webpack-dev-server":"1.16.2"
  }
}
```

  - \`npm install\` is used to configure dependencies written in package.json
  - \`app.js\` as a main js which will be responsible for calling all the other pages

```
import {printName} from './components/assg1.1_constant';
document.write("<h2>Hello from ES6 to connect to ES5</h2>");

/*ASSSIGNMENT 1*/
console.log("\nASSIGNMENT 1\n",printName());
```

  - After BABEL works on the ES6 script it gets converted to ES5 which is written in \`bundle.js\`. This configuration is done in \`webpack.config.js\`.

```
module.exports = {
  entry: "./app.js",
  output: {    filename: "bundle.js"   },
  devServer: {
    inline:true,
    port: 1234
  },
  module: {
       loaders: [
            {
                test: /\.js$/,
                exclude: /node_modules/,
                loader: 'babel',
                query: {
                    presets: ['es2015']
                }
            }
        ],
    },
  watch: true
}
```

  - Set the path for \`webpack server\` to build the dependencies that are needed to convert from ES6 to ES5 from \`./node_modules/.bin\`
  - Run \`webpack\` on console to convert entire ES6 code to ES5.

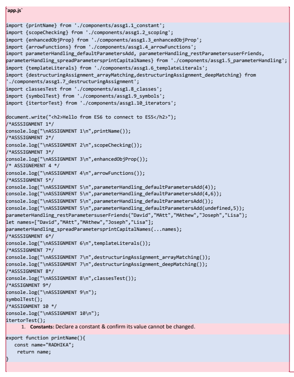
- Run `webpack-dev-server --inline` to start the server.
  - --inline: To automatically build once changes are made and display on browser.

- **Concepts**
  - **Constants**
  - **Scoping**
  - **Enhanced Object Properties**
    - **Computed Property Name**
    - **Method Properties**
  - **Object.assign()**
  - **Arrow Functions**
    - With arrow functions, code becomes compact and `'this' is accessible inside nested functions also.
  - **Extended Parameter Handling**
    - **Default Parameter Values**
    - **Rest Parameters** (`...` is given along as parameters)
    - **Spread parameters**
  - **Template Literals** (Backtick(``) are used to give multiline strings as well for displaying formatted strings)
  - **De-structuring Assignments**
    - **Array matching**
    - **Object matching**
      - **Shorthand notation**
      - **Deep matching**
      - **Parameter context**
    - **Fail-soft De-structuring**
  - **Modules**
    - Every js file is a module and it is restricted to that js file until and unless you import it.
    - 'Alias' is given while exporting and importing like column aliases in SQL
    - If 'default' keyword is used while exporting then no need to add {} while importing
    - Only one default export function is allowed per module.
    - Function called by default function need not be exported
    - `Default` is used for exporting main functions. i.e. the entry point
  - **Classes**
    - Class can have only one constructor(default or parameterized) and is declared using the keyword `constructor`
    - Need not declare attributes in class, it is injected using 'this'.
    - Getters and setters are allowed using 'get' and 'set' keywords
    - The 'get' and 'set' function will be having the attribute as the name of function which be set or get using this._'attributename'
      - Ex: if attribute name is `id`

```
get name(){
        return this._name;
}
set name(value){
        this._name=value;
}
```

  - **Inheritance**
    - Multiple inheritance not allowed.
    - 'super' keyword can be used in ES6
    - overloading of static functions is allowed using 'super' keyword

- **Symbols**
  - All Javascript objects are public
  - To create private objects, we use Symbol()
  - These private variables if needed by another function must be passed as parameters
  - There is absolutely no meaning to export a function if it takes symbol reference.
- **Iterators**
  - `for...of` loop in ES6 iterates through elements of array whereas `for...in` iterates through indices of array.
  - 'Symbol. Iterator' is a global symbol and can be used anywhere but not regenerated
  - Every array object has a key 'Symbol. Iterator' whose value is a function that allows to navigate through the array.
  - **CUSTOM ITERATOR**
    - Create object with key 'Symbol. Iterator' which will be a function that returns an object which has a single key 'next' which itself is a function which return two values [value, done] produced after writing the logic of iterator.
      Value: the next value to be returned
      Done: bool value indicating if it's the end of



`app.js`

```
import {printName} from './components/assg1.1_constant';
import {scopeChecking} from './components/assg1.2_scoping';
import {enhancedObjProp} from './components/assg1.3_enhancedObjProp';
import {arrowFunctions} from './components/assg1.4_arrowFunctions';
import parameterHandling_defaultParametersAdd, parameterHandling_restParametersuserFriends,
parameterHandling_spreadParametersprintCapitalNames} from './components/assg1.5_parameterHandling';
import {templateLiterals} from './components/assg1.6_templateLiterals';
import {destructuringAssignment_arrayMatching,destructuringAssignment_deepMatching} from
'./components/assg1.7_destructuringAssignment';
import classesTest from './components/assg1.8_classes';
import {symbolTest} from './components/assg1.9_symbols';
import {itertorTest} from './components/assg1.10_iterators';

document.write("<h2>Hello from ES6 to connect to ES5</h2>");
/*ASSSIGNMENT 1*/
console.log("\nASSIGNMENT 1\n",printName());
/*ASSSIGNMENT 2*/
console.log("\nASSIGNMENT 2\n",scopeChecking());
/*ASSSIGNMENT 3*/
console.log("\nASSIGNMENT 3\n",enhancedObjProp());
/* ASSIGNEMENT 4 */
console.log("\nASSIGNMENT 4\n",arrowFunctions());
/*ASSSIGNMENT 5*/
console.log("\nASSIGNMENT 5\n",parameterHandling_defaultParametersAdd(4));
console.log("\nASSIGNMENT 5\n",parameterHandling_defaultParametersAdd(4,6));
console.log("\nASSIGNMENT 5\n",parameterHandling_defaultParametersAdd());
console.log("\nASSIGNMENT 5\n",parameterHandling_defaultParametersAdd(undefined,5));
parameterHandling_restParametersuserFriends("David","MAtt","MAthew","Joseph","Lisa");
let names=["David","MAtt","MAthew","Joseph","Lisa"];
parameterHandling_spreadParametersprintCapitalNames(...names);
/*ASSSIGNMENT 6*/
console.log("\nASSIGNMENT 6\n",templateLiterals());
/*ASSSIGNMENT 7*/
console.log("\nASSIGNMENT 7\n",destructuringAssignment_arrayMatching());
console.log("\nASSIGNMENT 7\n",destructuringAssignment_deepMatching());
/*ASSSIGNMENT 8*/
console.log("\nASSIGNMENT 8\n",classesTest());
/*ASSIGNMENT 9*/
console.log("\nASSIGNMENT 9\n");
symbolTest();
/*ASSIGNMENT 10 */
console.log("\nASSIGNMENT 10\n");
itertorTest();
```

1. **Constants:** Declare a constant & confirm its value cannot be changed.

```
export function printName(){
   const name="RADHIKA";
    return name;
}
```

Figure 1: CODE SNIPPETS(DOUBLE CLICK TO EXPAND CODE)

Commented [RN1]: DOUBLE CLICK TO EXPAND CODE AND `CTRL+W` TO COME BACK TO THIS WINDOW.