# **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.



* + `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



* + 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`.



* + 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`

****

* + **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

****

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