LAB # 7

Modern Javascript (ES5\ES6\ESNEXT) and Babel

OBJECTIVE

To get familiar with modern Javascript and Babel, and their features.

THEORY

Javascript has gone through some changes and many good features which are available in other languages are added in it to circumvent the shortcomings it had. The changes were long awaited. For example, it did not have block level variable scope, concept of classes, module import and export syntax etc.

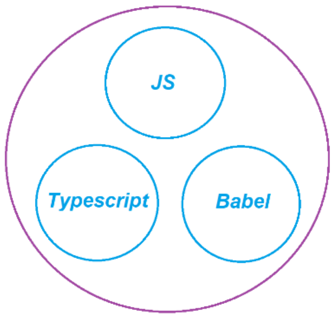
Although many features have been added into Javascript to make it as feature-rich and useful as possible, but there are still many things which developers around the world wish them to be available in it.

The purpose of *Babel* is to compile the code from one flavor of JS into another flavor of JS, and hence the term *Transpile*. That is, when the code is transformed from same language but with different flavor/syntax into another flavor/syntax of the same language, the process of compilation rather becomes *transpilation*.

The great benefit of using *Babel* is not only transpiling the code from one flavor to another but also it allows us to experiment with the *upcoming* JS features which have not yet been incorporated into existing JS engines.

Babel is not limited to transpiling the JS code, but it can transform any code/syntax into another syntax using its very important feature called *plugins*. Plugin system in Babel is used to make it extensible to also process other code formats than just JS. For example, there are CSS plugins, HTML plugins and so on. JS processing is the default format which does not require using any plugin.

We also have concept of *presets* in Babel besides *plugin.* The purpose of presets is to combine the related plugins in one collection and giving it a name, which can then be used to indicate to Babel to use all the plugins found under that specific preset. ReactJs and many other libraries provide us with many presets to transpile their code (which have their own syntax to write code instead of JS) into the one understood by the NodeJs and browser environments.



<u>Samples:</u>

1 *Modern JS – ES6 - 2015*: https://codepen.io/syed-owais-owais/pen/abWXWvx

Code:

```
es6Loops();
es6Functions();
es6BlockScope();
es6ObjectsManipulation();
es6Classes();
```

//------------------LOOP - ES6 - 2015 for-in vs for-of-----------------------------------

```
function es6Loops() {
   //Loop over array using for-in - This will print array index
   const studentGPAs = [3.5, 3, 4, 3.7];
   for (gpaIndex in studentGPAs) {
      console.log(gpaIndex);
   }

   //Loop over array using for-of - This will print array values
   for (gpaValue of studentGPAs) {
      console.log(gpaValue);
   }

   //Loop over object using for-in - This will print object keys. This loop cannot be used to iterate
over object values
   const student = {
      firstname: "abc",
      lastname: "xyz",
      email: "xyz@ssuet.com"
   };
```

```
    for (key in student) {
        console.log(key);
    }
}
```

//------------------LOOP - ES6 - Functions-----------------------------------

```
function es6Functions() {
    //Arrow based function with default parameter value
    const checkAssignment = (assignment = "read lab manuals") =>
        console.log(assignment);
    checkAssignment();

    //Use Promise 'then' method instead of callbacks function for async operations
    const lectureResponse = new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve("1 second lecture finished");
        }, 1000);
    });
    lectureResponse.then((response) => {
        console.log(response);
    });

    //Generator functions - One which return results multiple times as long as there are yield
statements. Note the syntax with *
    function* createLabsSchedule() {
        yield "Sep-15th-2021";
        yield "Sep-22nd-2021";
        yield "Sep-29th-2021";
        yield "Oct-06th-2021";
        yield "Oct-13th-2021";
        yield "Oct-20th-2021";
        yield "Oct-27th-2021";
    }

    // call it in a loop for every yielded value
    const labsSchedule = createLabsSchedule();
    for (labDate of labsSchedule) {
        console.log(labDate);
    }
    // or call it one by one
    const labsScheduleOneByOne = createLabsSchedule();

    console.log(labsScheduleOneByOne.next().value);
    console.log(labsScheduleOneByOne.next().value);
    console.log(labsScheduleOneByOne.next().value);
}
```

//------------------LOOP - ES6 - Block Level Scope----------------------------------

```
function es6BlockScope() {
    //function level scope
    var testScope = true;
    if (testScope) {
        //block level scope
        const blockLevelScope1 = true;
        let blockLevelScope2 = true;

        //function level scope
        var functionLevelScope1 = true;

        console.log(`inside block: const variable: ${blockLevelScope1}`);
        console.log(`inside block: let variable: ${blockLevelScope2}`);
        console.log(`inside block: var variable: ${functionLevelScope1}`);
    }

    try {
        console.log(blockLevelScope1);
    } catch {
        console.log(
            'variable declared with const is not available outside of block ("{}") scope'
        );
    }

    try {
        console.log(blockLevelScope2);
    } catch {
        console.log(
            'variable declared with let is not available outside of block ("{}") scope'
        );
    }

    console.log(functionLevelScope1);
}
```

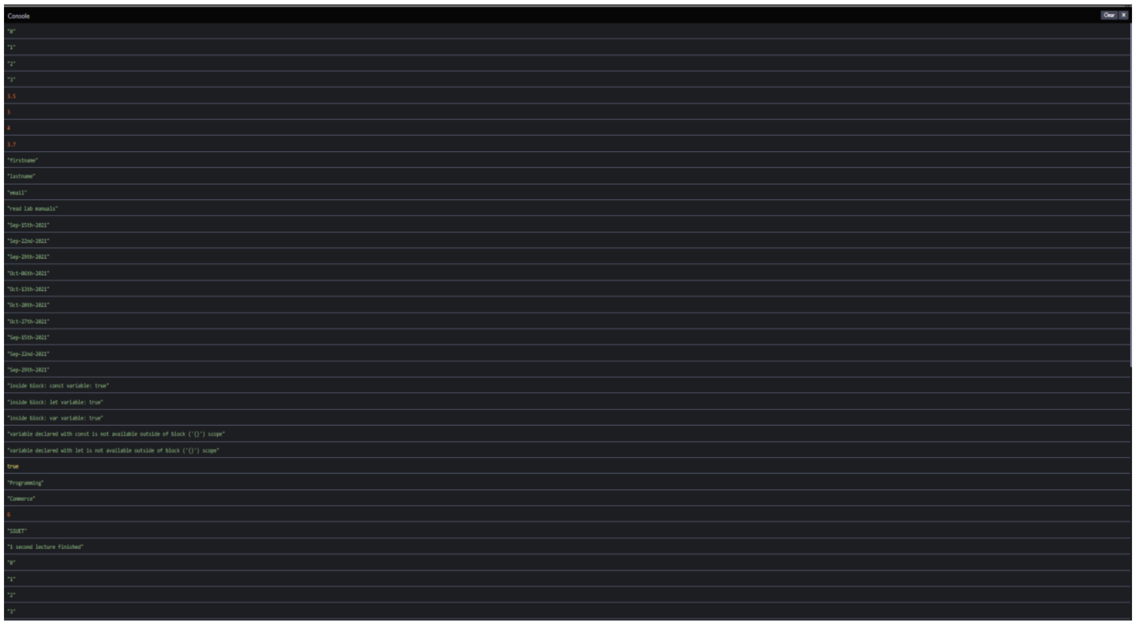
//------------------LOOP - ES6 - Objects Manipulation-----------------------------------

```
function es6ObjectsManipulation() {
    //Object creation
    const university = Object.create({
        name: "SSEUT",
        location: "Gulshan e Iqbal",
        foundedOn: new Date(1993, 1, 1),
        faculties: ["IT", "COMMERCE", "NANO-TECHNOLOGY"],
        open: () => {},
        close: () => {}
    });

    //Object destructuring - extracing properties into individual variables. The name and location
    //properties already exist on university object
    const { name, location } = university;

    //Object creation - using shorter syntax. Declaring the properties as name and location
    //and assigning values at the same time
    const universityCampus1 = {
        name,
        location
    };

    //Object destructuring - using rest syntax. First all university properties will be
    //copied, and then name and foundedOn properties will be overwritten inline if existed
    //otherwise, it will create new properties if not found on university object being destructured
    const universityCampus2 = {
        ...university,
        name: "SSUET - Campus 1",
        foundedOn: new Date(2000, 1, 1)
    };
}
```

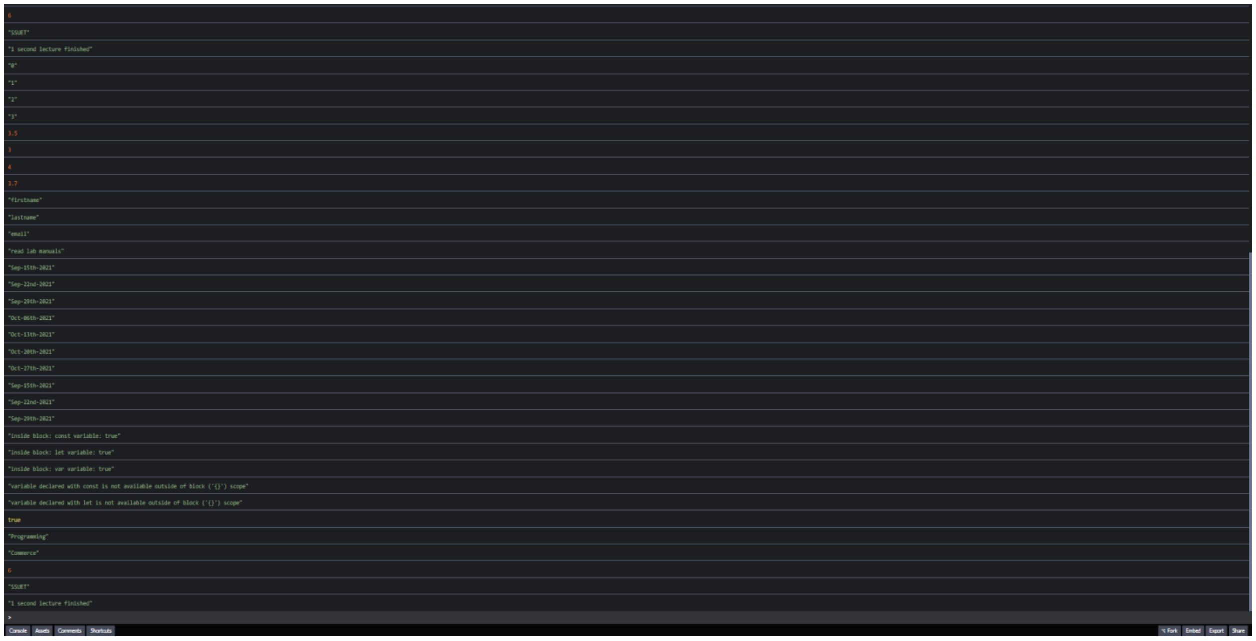
//------------------LOOP - ES6 - Classes-----------------------

```
function es6Classes() {
   //Modules
   class Presentation {
      constructor(date, topic, lead, duration) {
         this.date = date;
         this.topic = topic;
         this.lead = lead;
         this.duration = duration;
      }

      static getLocation() {
         return "SSUET";
      }
   }

   class PowerPointPresentation extends Presentation {
      constructor(noOfSlides) {
         super();
         this.noOfSlides = noOfSlides;
      }
   }
   const programmingPresentation = new Presentation(
      new Date(2021, 10, 10),
      "Programming",
      "XYZ",
      2
   );
   const commercePresentation = new Presentation(
      new Date(2021, 10, 10),
      "Commerce",
      "XYZ",
      3
   );
   const pptPresentation = new PowerPointPresentation(6);

   console.log(programmingPresentation.topic);
   console.log(commercePresentation.topic);
   console.log(pptPresentation.noOfSlides);
   console.log(Presentation.getLocation());
}
```

Output:



```
"0"
"1"
"2"
"3"
3.5
3
4
3.7
"firstname"
"lastname"
"email"
"read lab manuals"
"Sep-15th-2021"
"Sep-22nd-2021"
"Sep-29th-2021"
"Oct-06th-2021"
"Oct-13th-2021"
"Oct-20th-2021"
"Oct-27th-2021"
"Sep-15th-2021"
"Sep-22nd-2021"
"Sep-29th-2021"
"inside block: const variable: true"
"inside block: let variable: true"
"inside block: var variable: true"
"variable declared with const is not available outside of block ('{}') scope"
"variable declared with let is not available outside of block ('{}') scope"
true
"Programming"
"Commerce"
6
"SSUET"
"1 second lecture finished"
"0"
"1"
"2"
"3"
```



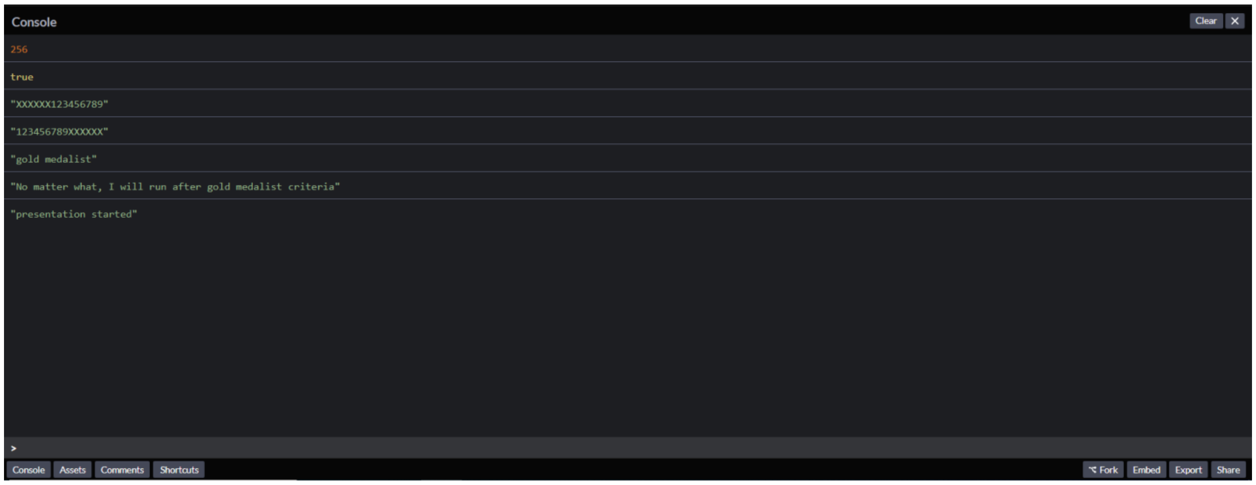
```
6
"SSUET"
"1 second lecture finished"
"0"
"1"
"2"
"3"
3.5
3
4
3.7
"firstname"
"lastname"
"email"
"read lab manuals"
"Sep-15th-2021"
"Sep-22nd-2021"
"Sep-29th-2021"
"Oct-06th-2021"
"Oct-13th-2021"
"Oct-20th-2021"
"Oct-27th-2021"
"Sep-15th-2021"
"Sep-22nd-2021"
"Sep-29th-2021"
"inside block: const variable: true"
"inside block: let variable: true"
"inside block: var variable: true"
"variable declared with const is not available outside of block ('{}') scope"
"variable declared with let is not available outside of block ('{}') scope"
true
"Programming"
"Commerce"
6
"SSUET"
"1 second lecture finished"
>
```

2 *Modern JS – ESNext:* https://codepen.io/syed-owais-owais/pen/YzVgQNQ

Code:

```
//All the ES updates add further features into the core JS language.
es7();
es8();
es9();

function es7() {
  //exponent operator **
  const base = 2;
  const exponent = 8;
  console.log(base ** exponent);

  //Array.includes
  const GPAList = [2, 3, 4, 3.5, 3.9];
  console.log(GPAList.includes(2));
}

async function es8() {
  //padStart and padEnd
  console.log("123456789".padStart(15, "X"));
  console.log("123456789".padEnd(15, "X"));

  //async/await function
  async function givePresentation() {
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        resolve("presentation started");
      }, 1000);
    });
  }

  const presentationStatus = await givePresentation();
  console.log(presentationStatus);
}

function es9() {
  const goldMedalistPromise = new Promise((resolve, reject) => {
    const GPA = 4.0;
    if (GPA < 4) {
      reject("no gold medal");
    } else {
      resolve("gold medalist");
    }
  });

  //finally
  goldMedalistPromise
    .then((result) => {
      console.log(result);
    })
    .catch((error) => {
      console.log(error);
    })
    .finally(() => {
      console.log("No matter what, I will run after gold medalist criteria");
    });
}
```

Output:



```
256
true
"XXXXXX123456789"
"123456789XXXXXX"
"gold medalist"
"No matter what, I will run after gold medalist criteria"
"presentation started"
```

3 *Babel*: https://codepen.io/syed-owais-owais/pen/YzVgJey

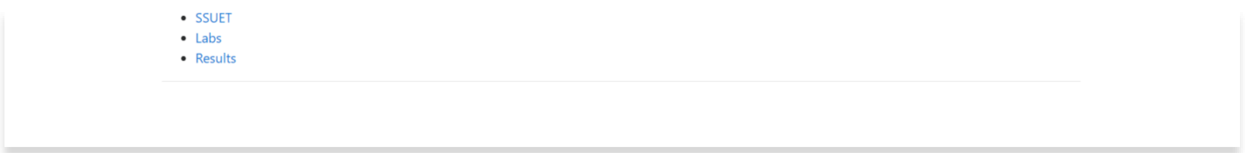
Code:

```
let { BrowserRouter, Link, Route} = ReactRouterDOM;
let Router = BrowserRouter;
```

```
// SSUET
class App extends React.Component {
  render() {
    return (
      <Router>
        <div className="container">
          <ul>
            <li>
              <Link to="/">SSUET</Link>
            </li>
            <li>
              <Link to="/labs">Labs</Link>
            </li>
            <li>
              <Link to="/results">Results</Link>
            </li>
          </ul>
          <hr />
          <Route exact path="/" component={SSUET} />
          <Route path="/labs" component={Labs} />
          <Route path="/results" component={Results} />
        </div>
      </Router>
    );
  }
}

const SSUET = () => (
  <div>
    <h1>SSUET</h1>
    <p class="lead">SSUET will be arranging labs for students to be taught by teachers</p>
  </div>
);
const Labs = () => (
  <div>
    <h1>Labs</h1>
    <p class="lead">Labs will be conducted by students</p>
  </div>
);
const Results = () => (
  <div>
    <h1>Results</h1>
    <p class="lead">Results will be prepared by teachers</p>
  </div>
);
```

```
ReactDOM.render(<App />, document.getElementById("ssuet"));
```

Output:



- SSUET
- Labs
- Results

Lab Task

1 *Modern JS*
   a. Create a class called *Library* and add few properties like *sections*, *books*, *manager*, and *openingTime* and *closingTime*.
   b. Add few functions like *manageLibrary, issueBooks, addNewSection, openLibrary,* and *closeLibrary*.
   c. Use arrow functions instead of regular functions for all function definition.
   d. Return promise from *issueBooks*, *openLibrary* and *closeLibrary* functions.
   e. Return books from issueBooks promise. Use *then* to receive the result.
   f. Return openingTime from openLibrary function. Use *setTimeout* and resolve after 2 seconds. Use *async/await* keywords pair to get the results.
   g. Return closingTime from closeLibrary function. Use *setTimeout* and resolve after 1 second. Use *async/await* keywords pair to get the results.
   h. Create instances of the *Library* class and call each function as mentioned above.
   i. Loop through the keys of each instance using for-in loop and log the keys.

2 *Babel*
   a. Create *Student* registration form using React.
   b. Create *Teacher* login form using React.
   c. Create two routes for each student and teacher forms and provide links to redirect to their respective forms.
   d. Render React forms using Babel transpiler.

Home Task

1 *Modern JS*
   a. Create a *generator* function called *generateRegistrationNumbers.*
      i. Accept the parameter asking the upper *limit* of the Registration number to be generated.
      ii. Calling the *next* method should return first registration number starting from 1
      iii. Calling the *next* method of generator again should return the next number.

iv. If the limit has been reached, the generator should stop giving next number no matter how many more times you call the *next* method of generator.