1. JavaScript Chapter

Exercise 1 (35 mins)

```
function validISBN10(isbn) {
    // TODO: return true if (and only if) isbn is a valid 10-digit ISBN.

    if(isbn.toString().length == 10){       //if the number has 10 digits go
and check if its valid
        sNumber = isbn.toString();
        var result = 0;
        var i = 0;

       while( i < isbn.toString().length){

        if( sNumber.charAt(i) === 'X'){
            result += 10 * (i+1) ;
        }else{
            result += sNumber.charAt(i) * (i+1) ;
        }
        i++;
       }

       if(result % 11 == 0)             //if mod is 0 then return true
        return true;
       else
        return false;                   //else return false
    }
    else
        return false;
   }
```

```
41    console.log("1112223339 -->  " + validISBN10('1112223339'))
42    console.log("111222333 --> " +validISBN10('111222333'))
43    console.log("1112223339X --> " +validISBN10('1112223339X'))
44    console.log("1234554321 --> " +validISBN10('1234554321'))
45    console.log("1234512345 --> " +validISBN10('1234512345'))
46    console.log("048665088X --> " +validISBN10('048665088X'))
47    console.log("X123456788 --> " +validISBN10('X123456788'))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS C:\Users\Chris\Documents\epanalipsi\js-basics> node index.js
1112223339 -->  true
111222333 --> false
1112223339X --> false
1234554321 --> true
1234512345 --> false
048665088X --> true
X123456788 --> true
PS C:\Users\Chris\Documents\epanalipsi\js-basics>
```

Exercise 2 (30 mins)

```
function allSquaredPairs(num) {
    const sqr1 = [];
    const sqr2 = [];
    var i = 0;
    if(num < 0)  // make integer a non-negative
        num = num * (-1);

    if(num >= 2147483647)  // max(num) === 2147483647
      return output;

    while(Math.pow(i, 2) <= num){ //adding values to two different arrays to
find the  pairs whose values - when squared - sum to the given integer
        sqr1.push(i);
        sqr2.push(i);
        i++;
    }
    let output = [];
    i = 0;
    while (i < sqr1.length ){
        var j=0;
       while ( j < sqr2.length){

        if (Math.pow(sqr1[i],2) + Math.pow(sqr2[j],2) === num){

            if(output.toString().includes(i)){          // Return every unique
pair of numbers [a,b] where (a * a) + (b * b) = num;
              break;
            }
             output.push([sqr1[i],sqr2[j]])
        }
        j++;
       }

        i++;
    }
    return output ;     // return value will be a two-dimensional array [[]]
   }
```

```
88      console.log(allSquaredPairs(0));
89      console.log(allSquaredPairs(1));
90      console.log(allSquaredPairs(2));
91      console.log(allSquaredPairs(3));
92      console.log(allSquaredPairs(4));
93      console.log(allSquaredPairs(5));
94      console.log(allSquaredPairs(25));
```

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

PS C:\Users\Chris\Documents\epanalipsi\js-basics> node index.js
[ [ 0, 0 ] ]
[ [ 0, 1 ] ]
[ [ 1, 1 ] ]
[]
[ [ 0, 2 ] ]
[ [ 1, 2 ] ]
[ [ 0, 5 ], [ 3, 4 ] ]
[ [ 1, 18 ], [ 6, 17 ], [ 10, 15 ] ]
[ [ 10, 30 ], [ 18, 26 ] ]
```

3. React.js Chapter (30 mins)

a. What is the difference between state and props?

Props are used to pass the data between one component from another (For parent-child communication).

State is referred to the local state of the component which cannot be accessed outside of the component and only can be used & modified inside the component.

b. **What are synthetic events in React? (Provide an example)?**

Synthetic Events are a cross-browser wrapper around the browser's native event. It has the same interface as the browser's native event, including preventDefault(),onClick(),onChange() etc .

Examples:

event.preventDefault() //Clicking on a "Submit" button, prevent it from submitting a form

```
function showAlert () {
    alert("Hello World!");
 }

 <button onClick={() => {showAlert()}}>show alert </button>
```

c. **What are portals in React?**

Portals render the child component outside the hierarchy of its parent component. For example, it allows you to keep child parent relation when it comes to click events and when it comes to rendering out your content in jsx, but you can actually render that content somewhere else by taking advantage of portals

e. **What will happen if you use setState in constructor?**

When you use setState(), it will cause react to rerender the component and all its children. Which you don't need in the constructor, since the component hasn't been rendered anyway.

Exercise React (1.5 hour)

```
//App.js
import { useState } from 'react';
import { useEffect } from "react";
import { List } from 'semantic-ui-react';

import './App.css';
function App() {

  const [records, setRecords] = useState(0);

  useEffect(() => { //on render call for useEffect and fetch the data from api
    fetch("https://jsonplaceholder.typicode.com/todos")
      .then((response) => response.json())
      .then((data) => {
        setRecords((data));  //insert the data in a state variable
      })


  }, [records]);

  return (
    <div className="App">
      <div className="Flex-box">
        <List style={{ display: "inline-grid" }}>
          {
            (records || []).map((item) => (
              <List.Item key={item.id}  >
                <br />
                {item.completed === true && (  //if its completed give it a
backround color of green
                  <li className="completed">  <br /> id: {item.id}
                  <br />
                  title: {item.title}</li>
                )}

                {item.completed === false && (

                  <li className="non-completed" >  <br />id: {item.id}
                  <br />
                  title: {item.title}</li>)}

              </List.Item>
            ))
          }
        </List>
      </div>
```

```
    </div>
  );
}

export default App;
```

```
/* App.css */

.App {
  text-align: center;

  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;

  color: white;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
```

```
}

.Flex-box {
  height: 95vh;
  display: flex;
  justify-content: center;
  overflow: auto;
  border: 10px solid #8d87878a;
  border-radius: 10px;
}

.completed {
  font-weight: bold;
  background: #1c6f0f;
  color: white;
  display: block;
  height: 100%;
  height: 90%;
  border: 2px solid #705959;
  border-radius: 10px;
  margin-top: 10px;
}

.non-completed {
  font-weight: bold;
  background: #962828;
  color: white;
  display: block;
  height: 100%;
  height: 90%;
  border: 2px solid #705959;
  border-radius: 10px;
  margin-top: 10px;
}
```