# Assignment-6

WAP of a function getUserData that ------ as an argument.

```
function getUserData(userId, callback) {
  setTimeout(() => {
    const userData = {id: userId, name: 'Kunal', email: 'Kunal@xyz.com'};
    callback(userData);
  }, 2000);
}
getUserData(1, function(data) {
  console.log(data);
});
```

O/P

id: 1, name: 'Kunal', email: 'Kunal@xyz.com'

Write a function fetchMultipleData ------ fetched data.

```
function fetchMultipleData(urls, callback) {
  const fetchPromises = urls.map(url => fetch(url).then(response => response.
                                                          json()));
  Promise.all(fetchPromises)
    .then(data => {
      callback(data);
    })
    .catch(error => {
      console.error('Error fetching data:', error);
    });
}

const urls = [
  'https://jsonplaceholder.typicode.com/posts/1',
  'https://jsonplaceholder.typicode.com/posts/2',
];
fetchMultipleData(urls, function(data) {
  conside.log(data);
});
```

03) Write a function fetchData ---------- error message.

Soln
```
function fetchData() {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            const success = true;
            if (success) {
                const data = {id: 1, name: 'Sample Data'};
                resolve(data);
            } else {
                reject('Error fetching data');
            }
        }, 2000);
    });
}

fetchData()
    .then(data => {
        console.log(data);
    })
    .catch(error => {
        console.error(error);
    });
```

04) Implement a fn calculateArea ---------- error message.

Soln
```
function calculateArea(length, width) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            if (typeof length === 'number' && typeof width === 'number' &&
                length > 0 && width > 0) {
                const area = length * width;
                resolve(area);
            } else {
                reject('Invalid dimensions');
            }
        }, 1000);
    });
}

calculateArea(5, 10)
    .then(area => {
        console.log('Area:', area);
    })
    .catch(error => {
        console.error(error);
    });
```

O/P
Area: 50
Invalid dimensions

5) Rewrite the fetchData function from Q3 using async/await syntax.

try

```
async function fetchData() {
  return new Promise ((resolve, reject) => {
    setTimeout(() => {
      const success = true;
      if (success) {
        const data = {id: 1, name: 'Sample Data'};
        resolve(data);
      } else {
        reject ('Error fetching data');
      }
    }, 2000);
  });

async function fetchDataAsync() {
  try {
    const data = await fetchData();
    console.log(data);
  } catch (error) {
    console.error(error);
  }
}
fetchDataAsync();
```

o/p

{id: 1, name: 'Sample Data'}

Error fetching data

Create an async function ..... data.

```
async function fetchAndProcessData() {
  try {
    const response = await fetch('https://..../posts');
    const data = await response.json();
    const processedData = data.filter(post => post.userId === 1);
    return processedData;
  } catch (error) {
    console.error('Error fetching or processing data:', error);
  }
}
```

```
async function displayProcencedData() {
    const data = await fetchAndProcessData();
    console.log(data);
}
displayProcencedData();
```

# Assignment - 7

1) Create a Simple Counter Component : ..... event handling.

```
import React, {useState} from 'react';
function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <h1> Counter: {count} </h1>
      <button onClick={() => setCount(count+1)}> Increment </button>
      <button onClick={() => setCount(count-1)}> Decrement </button>
    </div>
  );
}
function App(){
  return (
    <div>
      <Counter />
    </div>
  );
}
export default App;
```

2) Build a Todo List App : ..... user input

```
import React, {useState} from 'react';
function TodoList() {
  const [todos, setTodos] = useState([]);
  const [newTodo, setNewTodo] = useState('');
const addTodo = () => {
    if (newTodo.trim()) {
      setTodos([...todos, {text: newTodo, completed: false}]);
      setNewTodo('');
    }
};
  const toggleTodo = index => {
    const newTodos = todos.map((todo, i) => (
      i === index ? {...todo, completed: !todo.completed} : todo
    ));
    setTodos(newTodos);
  };
```

```
const deleteTodo = index => {
  const newTodos = todos.filter((_, i) => i !== index);
  setTodos(newTodos);
};
return (
  <div>
    <h1>Todo List</h1>
    <input
      type="text"
      value={newTodo}
      onChange={(e) => setNewTodo(e.target.value)}
      placeholder="Add a new task"
    />
    <button onClick={addTodo}>Add</button>
    <ul>
      {todos.map((todo, index) => (
        <li key={index} style={{ textDecoration: todo.completed ? 'line-t
                                                              : 'none' }}>
          <span onClick={() => toggleTodo(index)}>{todo.text}</span>
          <button onClick={() => deleteTodo(index)}>Delete</button>
        </li>
      ))}
    </ul>
  </div>
);
}
function App() {
  return (
    <div>
      <TodoList />
    </div>
  );
}
export default App;
```

Implement a Basic Calculator : ..... in React

```
import React, {useState} from 'react';
function Calculator() {
  const [input, setInput] = useState('');
  const [result, setResult] = useState('');
  const handleClick = value => {
    setInput(input + value);
  };
  const calculate = () => {
    try {
      setResult(eval(input));
    } catch {
      setResult('Error');
    }
  };
  const clear = () => {
    setInput('');
    setResult('');
  };
  return (
    <div>
      <h1>Calculator</h1>
      <input type="text" value={input} readOnly />
      <div>Result: {result}</div>
      <div>
        {['1', '2', '3', '+', '4', '5', '6', '-', '7', '8', '9', '*', '0', '.', '=', '/'].
          map(value => (
            <button key={value} onClick={() => value === '=' ? calculate() : handleClick(value)}>
              {value}
            </button>
          ))}
        <button onClick={clear}>Clear</button>
      </div>
    </div>
  );
}
```

```
function App() {
  return (
    <div>
      <Calculator />
    </div>
  );
}
export default App;
```

---

Q4) Develop a Weather App: ..... APIs in React.

Ans

```
import React, { useState } from 'react';
function Weather() {
  const [location, setLocation] = useState('');
  const [weather, setWeather] = useState(null);
  const fetchWeather = async () => {
    if (location.trim()) {
      try {
        const response = await fetch('https://api. ..... ');
        const data = await response.json();
        setWeather(data);
      } catch {
        alert('Failed to fetch weather data');
      }
    }
  };
  return (
    <div>
      <h1> Weather App </h1>
      <input
        type = "text"
        value = {location}
        onChange = {(e) => setLocation(e.target.value)}
        placeholder = "Enter location" />
      <button onClick = {fetchWeather}> Get Weather </button>
      {weather && (
        <div>
          <h2> {weather.name} </h2>
          <p> {weather.weather[0].description} </p>
          <p> Temperature: {Math.round(weather.main.temp - 273.15)}
              °C </p>
        </div>
      )}
    </div>
  );
}
```

```
function App() {
  return (
    <div>
      <Weather />
    </div>
  );
}
export default App;
```

Create a Simple Quiz Game: ..... interactions.

```
import React, { useState, useEffect } from 'react';

const questions = [
  {
    question: 'What is the capital of France?',
    options: ['Paris', 'London', 'Rome', 'Berlin'],
    answer: 'Paris'
  },
  {
    question: 'Who wrote "Hamlet"?',
    options: ['Shakespeare', 'Dickens', 'Hemingway', 'Tolkien'],
    answer: 'Shakespeare'
  },
];

function Quiz() {
  const [currentQuestion, setCurrentQuestion] = useState(0);
  const [score, setScore] = useState(0);
  const [timeLeft, setTimeLeft] = useState(10);

useEffect(() => {
  if (timeLeft > 0) {
    const timer = setTimeout(() => setTimeLeft(timeLeft - 1), 1000);
    return () => clearTimeout(timer);
  } else {
    nextQuestion();
  }
}, [timeLeft]);

const nextQuestion = () => {
  if (currentQuestion < questions.length - 1) {
    setCurrentQuestion(currentQuestion + 1);
    setTimeLeft(10);
```

```
} else {
  alert('Quiz finished! Your score: ${score}');
}
};

const selectOption = option => {
  if (option === questions[currentQuestion].answer) {
    setScore(score + 1);
  }
  nextQuestion();
};

return (
  <div>
    <h1>Quiz Game</h1>
    <p>Time Left: {timeLeft} seconds</p>
    <h2>{questions[currentQuestion].question}</h2>
    {questions[currentQuestion].options.map(option => (
      <button key={option} onClick={() => selectOption(option)}>
        {option}</button>
    ))}
    <p>Score: {score}</p>
  </div>
);
}

function App() {
  return (
    <div>
      <Quiz />
    </div>
  );
}

export default App;
```