```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>

<body>
    <input type="text" name="" id="event">
    <script src="event.js"></script>
</body>

</html>
```

This is a simple HTML document with an input field of type text. The JavaScript code is included from an external file called "event.js."

Now, let's look at the contents of "event.js":

```javascript

let a = document.getElementById("event");

// Event listener for click event
a.addEventListener("click", function (e) {
  console.log(e);
});

// Event listener for click event with more information about the
event
a.addEventListener("click", function (e) {
  console.log(e.type);       // Output: click
  console.log(e.shiftKey);   // Output: false (Shift key pressed or
not)
  console.log(e.target);     // Output: the element that triggered
the event (in this case, the input element)
});

// Event listener for input event
```

```
a.addEventListener("input", function (e) {
  console.log(e.target.value);  // Output: the current value of the
input field when it changes
});

// Event listener for click event with mouse coordinates
a.addEventListener("click", function (e) {
  console.log(e.clientX, e.clientY);  // Output: X and Y coordinates
of the mouse pointer when the click event occurs
});
```

Explanation:

1] Event Registration:

- The JavaScript code selects the input element with the id “event” using document.getElementById(“event”) and assigns it to the variable a.

2] Click Event:

- The first event listener logs the entire event object to the console when a click event occurs on the input field.

3] Click Event with Details:

- The second event listener for the click event logs specific details of the event:
- e.type: Outputs the type of the event ("click").
- e.shiftKey: Outputs whether the Shift key is pressed (true or false).
- e.target: Outputs the element that triggered the event (in this case, the input element).

4] Input Event:

- The third event listener is for the input event, which logs the current value of the input field whenever it changes.

5] Click Event with Mouse Coordinates:

- The fourth event listener for the click event logs the X and Y coordinates of the mouse pointer when the click event occurs.

These event listeners demonstrate various aspects of handling events in JavaScript, including accessing event properties and details.

Var:

- it supports re-declare, re-initialize
- it supports in function and block scope initialized.

Let

- it not supports re-declare, but supports re-initialize
- it supports in block scope

const

- it not supports re-declare, re-initialize
- it supports in block scope

JavaScript Hoisting

- Hoisting is JavaScript's default behavior of moving declarations to the top.

JavaScript Declarations are Hoisted

- In JavaScript, a variable or function can be declared after it has been used.
- In other words; a variable or function can be used before it has been declared.

The let and const Keywords

- Variables defined with let and const are hoisted to the top of the block, but not initialized.

```
var a = "hello";
var a = "world";
console.log(a);

let a = "hello";
a = "world";
console.log(a);

const a = "hi";
const a = "world";
console.log(a);
```

```
function name() {
  if (true) {
    var a = 12;
  }
  console.log(a);
}
name();

//let
function name() {
  if (true) {
    let a = 12;
    console.log(a);
  }
  //console.log(a) a is not defined.
}
name();

hoisting
hello();
function hello() {
  console.log("hi");
}

var a;
console.log(a); //unindefined
a = 5;

var a;
a = 5;
console.log(a); //5

a = 5;
console.log(a); //5
var a;
```

JavaScript Assignment 9 Question

Q1. W.A.P to display one card in light mode and dark mode, this includingbgColor, image, heading, paragraph, border, box shadow?

Answer:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Card Display</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            background-color: #f8f9fa;
            /* Default light mode background color */
            color: #212529;
            /* Default light mode text color */
            transition: background-color 0.5s, color 0.5s;
        }

        .card {
            background-color: #fff;
            /* Default light mode card background color */
            color: #212529;
            /* Default light mode card text color */
            border: 1px solid #dee2e6;
            /* Default light mode card border color */
            border-radius: 8px;
            padding: 20px;
```

```css
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    /* Default light mode box shadow */
    transition: background-color 0.5s, color 0.5s, border 0.5s, box-shadow 0.5s;
}

.dark-mode {
    background-color: #343a40;
    /* Dark mode background color */
    color: #dee2e6;
    /* Dark mode text color */
    border: 1px solid #495057;
    /* Dark mode card border color */
    box-shadow: 0 4px 8px rgba(255, 255, 255, 0.1);
    /* Dark mode box shadow */
}

img {
    max-width: 100%;
    height: auto;
    border-radius: 4px;
    margin-bottom: 15px;
}

h2 {
    font-size: 24px;
    margin-bottom: 10px;
}

p {
    font-size: 16px;
    margin-bottom: 0;
}

button {
    cursor: pointer;
    padding: 10px 20px;
    font-size: 16px;
    border: none;
    border-radius: 4px;
    background-color: #007bff;
    color: #fff;
    transition: background-color 0.3s;
```

```html
        }

        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>

<body>

    <div class="card" id="card">
        <img src="https://placekitten.com/300/200" alt="Card Image">
        <h2>Card Heading</h2>
        <p>This is a sample card paragraph. You can customize the content here.</p>
        <button onclick="toggleDarkMode()">Toggle Dark Mode</button>
    </div>

    <script src="assigment9.js">

    </script>

</body>

</html>
```

```javascript
function toggleDarkMode() {
  const body = document.body;
  const card = document.getElementById("card");

  body.classList.toggle("dark-mode");
  card.classList.toggle("dark-mode");
}
```

Q2. Explain the difference between var, let, const keywords with examples?

Answer:

In JavaScript, var, let, and const are keywords used to declare variables. However, they have some key differences in terms of scope, hoisting, and reassignment.

1] Var :

a) Variables declared with var are function-scoped, meaning they are only accessible within the function where they are declared.

b) var declarations are hoisted to the top of their scope, which means they are moved to the top during the execution phase, and you can use the variable before it's declared in the code.

c)var allows redeclaration and reassignment.

```
function exampleVar() {
  if (true) {
    var x = 10;
    console.log(x); // Outputs 10
  }
  console.log(x); // Outputs 10
}
exampleVar();
```

2] let:

a) Variables declared with let are block-scoped, meaning they are only accessible within the block ({}) where they are defined.

b) let declarations are also hoisted but not initialized. This means you cannot access the variable before the declaration.

c) let allows reassignment but not redeclaration in the same scope.

```
function exampleLet() {
  if (true) {
    let y = 20;
    console.log(y); // Outputs 20
  }
  // console.log(y); // Error: y is not defined (outside the block)
```

3] const:

a) Variables declared with const are block-scoped like let.

b) const variables cannot be reassigned once they are initialized. They must be assigned a value at the time of declaration.

c) const does not allow redeclaration in the same scope.

```
function exampleConst() {
  const z = 30;
  // z = 40; // Error: Assignment to a constant variable
  console.log(z); // Outputs 30
}
exampleConst();
```