REVISION

Page No :
Date :

1) What is an event in JS ?

In JavaScript, an event is an action or occurence that happens in the browser, which the browser can detect and respond to. These events are typically triggered by the user's interaction with the web page such as clicking a button, hovering over an element, typically on the keyboard, or even loading a page.

Common types of events in JS include:

1) Mouse Events:
   - click
   - mouseover
   - mouseout
   - dblclick

2) Keyboard Events:
   - keydown
   - keyup
   - keypress

3) Form Events:
   - submit
   - change
   - focus
   - blur

4) Window Events:
   - load
   - resize
   - scroll

Page No :
Date :

Document Events:
- DOMContentLoaded

How to use events in JS:

```
document.getElementById("myButton").addEventListener("click", function() {
    alert("Button was clicked!");
});
```

In this example, when the button with the ID "myButton" is clicked, an alert box will pop up with the message "Button was clicked!".

2) How do you add an EventListener to an element?

To add an EventListener to an element, you use the `addEventListener()` method. This method allows you to define what should happen when a specific event occurs on the targeted element.

Page No :
Date :

Syntax:

element.addEventListener(event, function, useCapture);

- element : The DOM element you want to add the event listener to.
- event : The type of event to listen for, such as "click", "mouseover", ...
- function : The function to execute when the event is triggered.
- useCapture : A boolean value that indicates whether the event should be captured or bubbled.

Example : Adding a click event listener:

```
// select the element
const myButton = document.getElementById
                 ("myButton");

// Add an event listener for the
'click' event
myButton.addEventListener("click", function()
{
    alert("Button was clicked!");
});
```

Page No :
Date :

Example 2: Using a Named Function.

```
// Define the function
function handleClick() {
    alert("Button was clicked!");
}
// Select the element
const myButton = document.getElementById("myButton");
// Add event listener
myButton.addEventListener("click", handleClick);
```

3) How can you remove an event listener?

To remove an event listener, you use the 'removeEventListener()' method. This method works in a similar way to 'addEventListener()' but it removes the event listener instead of adding it.

Syntax:

```
element.removeEventListener(event, function, useCapture);
```

Page No :
Date :

4) Difference b/w addEventListener & onclick Property

addEventListener	onclick Property
=> allows you to attach multiple event handlers to the same event on the same element. Each handler will be called in the order they were added.	=> Only allows one event handler at a time.
=> Supports both capturing and bubbling phases.	=> only works during the bubbling phase.

5) Explain event delegation & how it works?

Event delegation is a pattern based upon the concept of event bubbling. It is an event-handling pattern that allows you to handle events at a higher level in the

Page No :
Date :

DOM tree other than the level where the event was first received.
Event delegation is a technique in JS, where you take advantage of event bubbling to handle events at a higher level in the DOM, rather than attaching event listeners directly to the individual elements.

How it works:
1) Event Bubbling: When an event is triggered on an element, it starts from that element and bubbles up through the ancestors of that element in the DOM hierarchy until it reaches the root (<html>). During this process, any event handlers attached to these ancestors can respond to the event.

2) Event Delegation: Instead of

Page No :
Date :

attaching event listeners to each child element, you attach a single event listener to a parent element. The parent element listens for the event to bubble up from its children, and then you can determine which child element triggered the event.

6) What is DOM?

The Document Object Model is a programming interface for web documents. It represents the structure of an HTML or XML document as a tree of objects, where each object corresponds to a part of the document.

The DOM allows the programmer to change the structure, style, and content of a webpage. The DOM represents a document as a logical tree of nodes and objects, which

Page No :
Date :

allows Programming languages to interact with the page. You can use DOM to add, remove, or alter ~~contents in~~ elements in the document, such as changing the text content or setting attributes.

7) How do you create and append a new DOM element?

To create and append a new DOM element:

1) Create the new element using the 'document.createElement()' method.
2) Set any desired properties or attributes on the element.
3) Append the element to an existing element in the DOM using methods like 'appendChild()' or 'append()'.

Example:

Page No :
Date :

Html structure :

```
<body>
    <div id = "content">
        <h2> Existing content </h2>
    </div>
    <script src = script.js "> </script>
</body>
```

script.js

```
// create a new <p> element.
const newPara = document.createElement("p");

// set the text content of new element.
newPara.textContent = "This is a new added
                       para.";

// Append the new element to existing element.
const contentDiv = document.getElementById
                   ("content");
contentDiv.appendChild(newPara);
```

Page No :
Date :

8) How do you change the content of an HTML element?

You can change the content of an HTML element ~~and its~~ is several ways using JS. The most common methods are by manipulating the 'textContent', 'innerHTML', or 'innerText' properties of the element.

`value`: Use this to change the value of form elements like <input> or <textarea>.

9) What is an event listener & event handler?

=> An event listener is ~~an~~ a function or object that waits for a specific event to occur on a particular element and then executes a callback

function when that event happens.

```
// select the element
const myButton = document.getElementById("myButton");
// define the event listener
function handleClick() {
    alert("Button clicked!");
}
// attach event listener to the button.
myButton.addEventListener("click", handleClick);
```

⇒ Event Handler

is the function that executes when the event occurs. It is the callback function that provide to the event listener. The event handler defines what should happen in response to the event.

Page No :
Date :

10) How can we prevent the default behaviour of an element?

To prevent the default behavior of an element in JS, you use the `event.preventDefault()` method within an event handler.

```
function handleFormSubmit(event) {
  event.preventDefault();
  alert("Form submission prevented");
}
```