

Tutorial : Event Driven Programming

Event-driven programming

Events and event handlers

An event is an action, usually caused by a user, that the web browser responds to. Ex: A mouse movement, a key press, or a network response from a web server. Typically, the occurrence and timing of an event are unpredictable, since the user or web server can perform an action at any time.

Event-driven programming is a programming style where code runs only in response to various events. Code that runs in response to an event is called an event handler or event listener.

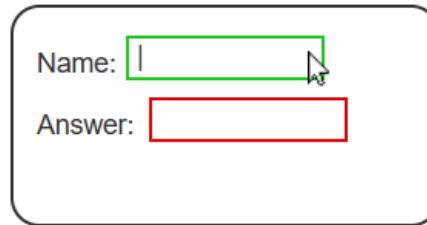
The web browser supports event-driven programming to simplify handling the many events a webpage must process. When an event happens, the browser calls the event's specified handlers. The web browser internally implements the code for detecting events and executing event handlers.

The example below modifies an input's `style` property, which sets the element's inline CSS styles. The input's border color changes colors when the input receives the focus or when focus is removed.

Example : Focus and Blur Event Handling

1. User clicks in the Name input box. Browser calls the input element's focus event handler, which changes the element's style. Browser then gives focus to input box.

```
<p>
  <label for="name">Name:</label>
  <input type="text" id="name">
</p>
<p>
  <label for="answer">Answer:</label>
  <input type="number" id="answer">
</p>
```



Name:

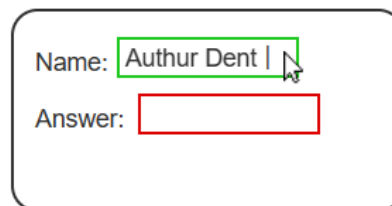
Answer:

```
let inputs = document.querySelectorAll("input");

for (let i = 0; i < inputs.length; i++) {
  let input = inputs[i];
  input.style.border = "1px solid red";
  input.addEventListener("focus", function() {
    this.style.border = "1px solid green";
  });
  input.addEventListener("blur", function() {
    this.style.border = "1px solid blue";
  });
}
```

2. User key presses are sent to Name input box.

```
<p>
  <label for="name">Name:</label>
  <input type="text" id="name">
</p>
<p>
  <label for="answer">Answer:</label>
  <input type="number" id="answer">
</p>
```



Name:

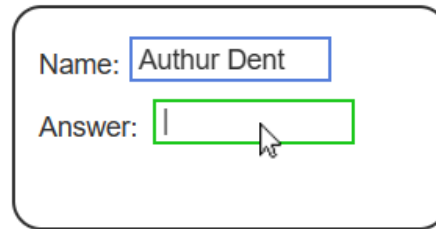
Answer:

```
let inputs = document.querySelectorAll("input");

for (let i = 0; i < inputs.length; i++) {
  let input = inputs[i];
  input.style.border = "1px solid red";
  input.addEventListener("focus", function() {
    this.style.border = "1px solid green";
  });
  input.addEventListener("blur", function() {
    this.style.border = "1px solid blue";
  });
}
```

3. User clicks the Answer input box. Browser calls the Name element's blur event handler, then calls the Answer element's focus handler, and then gives focus to the Answer input box.

```
<p>
  <label for="name">Name:</label>
  <input type="text" id="name">
</p>
<p>
  <label for="answer">Answer:</label>
  <input type="number" id="answer">
</p>
```



Name:

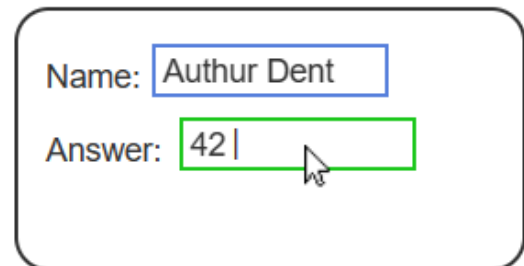
Answer:

```
let inputs = document.querySelectorAll("input");

for (let i = 0; i < inputs.length; i++) {
  let input = inputs[i];
  input.style.border = "1px solid red";
  input.addEventListener("focus", function() {
    this.style.border = "1px solid green";
  });
  input.addEventListener("blur", function() {
    this.style.border = "1px solid blue";
  });
}
```

4. User key presses are sent to Answer input box.

```
<p>
  <label for="name">Name:</label>
  <input type="text" id="name">
</p>
<p>
  <label for="answer">Answer:</label>
  <input type="number" id="answer">
</p>
```



Name:

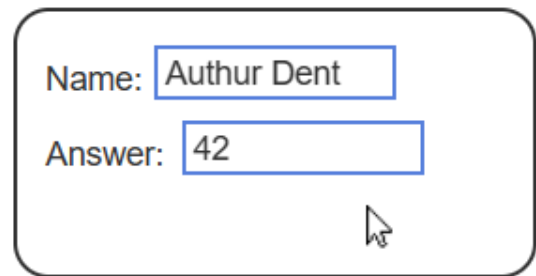
Answer:

```
let inputs = document.querySelectorAll("input");

for (let i = 0; i < inputs.length; i++) {
  let input = inputs[i];
  input.style.border = "1px solid red";
  input.addEventListener("focus", function() {
    this.style.border = "1px solid green";
  });
  input.addEventListener("blur", function() {
    this.style.border = "1px solid blue";
  });
}
```

5. When the user clicks elsewhere, the browser calls the Answer blur event handler.

```
<p>
  <label for="name">Name:</label>
  <input type="text" id="name">
</p>
<p>
  <label for="answer">Answer:</label>
  <input type="number" id="answer">
</p>
```



```
let inputs = document.querySelectorAll("input");

for (let i = 0; i < inputs.length; i++) {
  let input = inputs[i];
  input.style.border = "1px solid red";
  input.addEventListener("focus", function() {
    this.style.border = "1px solid green";
  });
  input.addEventListener("blur", function() {
    this.style.border = "1px solid blue";
  });
}
```

Common events

Each event is given a name that represents the corresponding action. Ex: The event name for a mouse movement is `mousemove`, and the event name for a key down is `keydown`.

click	<u>Caused by a mouse click.</u> The browser creates a click event when the user does a button press.
mouseover	<u>Caused by mouse entering the area defined by an HTML element.</u> A mouseover event occurs when the mouse moves over the HTML element area from outside the area.
mouseout	<u>Caused by mouse exiting the area defined by an HTML element.</u> A mouseout event occurs when the mouse moves from inside the HTML element area to an area outside that HTML element.
mousemove	<u>Caused by mouse moving within the area defined by an HTML element.</u> A mousemove event occurs when the mouse moves while remaining in the HTML element area.
keydown	<u>Caused by the user pushing down a key on the keyboard.</u> A keydown event occurs when the key is pressed. A separate event occurs when the key is released.
keyup	<u>Caused by the user releasing a key on the keyboard.</u> A keyup event occurs when the key is released. A separate event occurs when the key is pressed.

The following are events for which web developers commonly write handlers:

- A **change** event is caused by an element value being modified. Ex: Selecting an item in a radio button group causes a change event.
- An **input** event is caused when the value of an input or textarea element is changed.
- A **load** event is caused when the browser completes loading a resource and dependent resources. Usually load is used with the body element to execute code once all the webpage's CSS, JavaScript, images, etc. have finished loading.
- A **DOM ContentLoaded** event is caused when the HTML file has been loaded and parsed, although other related resources such as CSS, JavaScript, and image files may not yet be loaded.
- A **focus** event is caused when an element becomes the current receiver of keyboard input. Ex: Clicking in an input field causes a focus event.
- A **blur** event is caused when an element loses focus and the element will no longer receive future keyboard input.
- A **submit** event is caused when the user submits a form to the web server.

Registering event handlers

Handlers are written in three ways:

1. Embedding the handler as part of the HTML. Ex: `<button onclick="clickHandler()">Click Me</button>` sets the click event handler for the button element by using the `onclick` attribute. The attribute name used to register the handler adds the prefix "on" to the event name. Ex: The attribute for a mousemove event is `onmousemove`. Embedding a handler in HTML mixes content and functionality and thus should be avoided whenever possible.
2. Setting the DOM node event handler property directly using JavaScript. Ex: `document.querySelector("#myButton").onclick = clickHandler` sets the click event handler for the element with an id of "myButton" by overwriting the `onclick` JavaScript property. Using DOM node properties is better than embedding handlers within the HTML but has the disadvantage that setting the property only allows one handler for that element to be registered.
3. Using the JavaScript `addEventListener()` method to register an event handler for a DOM object. Ex: `document.querySelector("#myButton").addEventListener("click", clickHandler)` registers a click event handler for the element with the id "myButton". Good practice is to use the `addEventListener()` method whenever possible, rather than using element attributes or overwriting JavaScript properties. The `addEventListener()` method allows for separation of content and functionality and allows multiple handlers to be registered with an element for the same event.

Every handler has an optional event object parameter that provides details of the event. Ex: For a keyup event, the event object indicates which key was pressed and released, or for a click event, which element was clicked.

In the animation below, `keyupHandler()` uses `event.target` to access the text box object where the keyup event occurred. Inside an event handler, the `this` keyword refers to the element to which the handler is attached. So `event.target` and `this` both refer to the text box object in the event handler.

Example :

1. The window's `addEventListener()` method registers the handler `loadedHandler()` for the `DOMContentLoaded` event.

```
<!DOCTYPE html>
<html>
  <title>Keyup Demo</title>
  <script>
    window.addEventListener("DOMContentLoaded", loadedHandler);

    function loadedHandler() {
      let textBox = document.querySelector("#name");
      textBox.addEventListener("keyup", keyupHandler);
    }

    function keyupHandler(event) {
      if (event.key == "Enter") {
        let textBox = event.target;
        alert("Hello, " + textBox.value + "!");
      }
    }
  </script>
</body>
</html>
```

Name?

The window's `addEventListener()` method registers the handler `loadedHandler()` for the `DOMContentLoaded` event.

2. After the rest of the HTML is loaded and parsed, the `DOMContentLoaded` event occurs, and `loadedHandler()` is called.

```
<!DOCTYPE html>
<html>
  <title>Keyup Demo</title>
  <script>
    window.addEventListener("DOMContentLoaded", loadedHandler);

    function loadedHandler() {
      let textBox = document.querySelector("#name");
      textBox.addEventListener("keyup", keyupHandler);
    }

    function keyupHandler(event) {
      if (event.key == "Enter") {
        let textBox = event.target;
        alert("Hello, " + textBox.value + "!");
      }
    }
  </script>
</body>
</html>
```

Name?

3. The text box's `addEventListener()` method registers the handler `keyupHandler()` for the `keyup`

event.

```
<!DOCTYPE html>
<html>
  <title>Keyup Demo</title>
  <script>

    window.addEventListener("DOMContentLoaded", loadedHandler);

    function loadedHandler() {
      let textBox = document.querySelector("#name");
      textBox.addEventListener("keyup", keyupHandler);
    }

    function keyupHandler(event) {
      if (event.key == "Enter") {
        let textBox = event.target;
        alert("Hello, " + textBox.value + "!");
      }
    }

  </script>
</body>
  <label for="id">Name?</label>
  <input type="text" id="name">
</body>
</html>
```

Name?

4. When the user types the first letter, a keyup event occurs, which results in keyupHandler() being called.

```
<!DOCTYPE html>
<html>
  <title>Keyup Demo</title>
  <script>

    window.addEventListener("DOMContentLoaded", loadedHandler);

    function loadedHandler() {
      let textBox = document.querySelector("#name");
      textBox.addEventListener("keyup", keyupHandler);
    }

    function keyupHandler(event) {
      if (event.key == "Enter") {
        let textBox = event.target;
        alert("Hello, " + textBox.value + "!");
      }
    }

  </script>
</body>
  <label for="id">Name?</label>
  <input type="text" id="name">
</body>
</html>
```

Name?

5. The event.key is a string representing the pressed key ("P" for key P).


```

<!DOCTYPE html>
<html>
  <title>Keyup Demo</title>
  <script>

window.addEventListener("DOMContentLoaded", loadedHandler);

function loadedHandler() {
  let textBox = document.querySelector("#name");
  textBox.addEventListener("keyup", keyupHandler);
}

function keyupHandler(event) {
  if (event.key == "Enter") {
    let textBox = event.target;
    alert("Hello, " + textBox.value + "!");
  }
}

  </script>
  <body>
    <label for="id">Name?</label>
    <input type="text" id="name">
  </body>
</html>

```

Name?

6. Each keyup causes keyupHandler() to execute. When the user presses Enter, event.key is "Enter", and the if statement is true.

```

<!DOCTYPE html>
<html>
  <title>Keyup Demo</title>
  <script>

window.addEventListener("DOMContentLoaded", loadedHandler);

function loadedHandler() {
  let textBox = document.querySelector("#name");
  textBox.addEventListener("keyup", keyupHandler);
}

function keyupHandler(event) {
  if (event.key == "Enter") {
    let textBox = event.target;
    alert("Hello, " + textBox.value + "!");
  }
}

  </script>
  <body>
    <label for="id">Name?</label>
    <input type="text" id="name">
  </body>
</html>

```

Name?

7. event.target is the text box object that caused the keyup event. An alert dialog displays "Hello, Pam!"

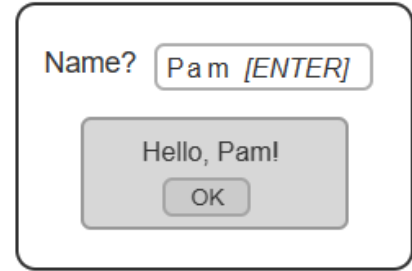
```
<!DOCTYPE html>
<html>
  <title>Keyup Demo</title>
  <script>

window.addEventListener("DOMContentLoaded", loadedHandler);

function loadedHandler() {
  let textBox = document.querySelector("#name");
  textBox.addEventListener("keyup", keyupHandler);
}

function keyupHandler(event) {
  if (event.key == "Enter") {
    let textBox = event.target;
    alert("Hello, " + textBox.value + "!");
  }
}

  </script>
  <body>
    <label for="id">Name?</label>
    <input type="text" id="name">
  </body>
</html>
```



Name? Pam [ENTER]

Hello, Pam!

OK