JavaScript - Keyboard Events

The keyboard events in JavaScript provide a way to interact with a web page or application based on the user's keyboard input. These events allow developers to capture and respond to various keyboard actions, such as key presses, key releases, and character inputs. The primary keyboard events in JavaScript include keydown, keypress, and keyup.

Common Keyboard Events

Keydown Event − When a key on the keyboard is pressed down, it triggers the keydown event. This event equips developers with information about the specific key that was pressed: this includes its code and an indicator of whether certain modifier keys such as Shift, Ctrl, or Alt were also depressed.

Keypress Event − The keypress event triggers when a user types an actual character. Non-character keys, such as Shift or Ctrl, do not activate this event. Developers frequently utilize it to capture user input for form fields or create interactive typing features.

Keyup Event − Upon the release of a previously pressed key, the system initiates the firing of a keyup event; this particular event proves beneficial in tracking specific keys' releases and subsequently implementing actions, thereby creating an interactive user experience.

Keyboard Event Properties

For keyboard events in JavaScript, several properties are commonly used to gather information about the key pressed. Here are some key properties specifically relevant to keyboard events −

Property	Description
event.key	String representing the key value of the pressed key.
event.code	String representing the physical key on the keyboard.
event.location	Integer indicating the location of the key on the keyboard.

event.ctrlKey	Boolean indicating whether the Ctrl key was held down.
event.shiftKey	Boolean indicating whether the Shift key was held down.
event.altKey	Boolean indicating whether the Alt key was held down.
event.metaKey	Boolean indicating whether the Meta (Command) key was held down.
event.repeat	Boolean indicating whether the key event is a repeat event.
event.isComposing	Boolean indicating whether the event is part of a composition of multiple keystrokes.
event.which	Deprecated property; previously used to identify the numeric key code.
event.getModifierState(keyArg)	Method that returns a boolean indicating whether the modifier key is pressed.

Example: Keydown Event

This example illustrates the application of JavaScript's keydown event. The event listener seizes the keydown event upon pressing any key, displaying in an HTML element identified as "output" - its corresponding key (an event property).

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
   <h3>Press any key</h3>
   <script>
      document.addEventListener('keydown', function (event) {
         document.getElementById('output').innerHTML =
         "Key pressed: " + event.key;
      });
   </script>
```

```
   <div id="output"></div>
</body>
</html>
```

Example: Keypress Event

In this example, the keypress event is utilized to capture a typed character. When a character is typed, the event listener triggers, and the character is displayed in the HTML element with the id "output".

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
   <h3>Type a character</h3>
   <div id="output"></div>
   <script>
      document.addEventListener('keypress', function (event) {
         document.getElementById('output').innerHTML =
         "Character pressed: " + event.key;
      });
   </script>
</body>
</html>
```

Example: Keyup Event

The keyup event is showcased in this example. It captures the event when a key is released after being pressed. The released key is then displayed on screen.

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
   <h3>Press and Release a key</h3>
   <div id="output"></div>
   <script>
      document.addEventListener('keyup', function (event) {
         document.getElementById('output').innerHTML =
```

```
            "Key released: " + event.key;
         });
      </script>
   </body>
   </html>
```

There is a difference between keydown and keypress. keydown is triggered when any key is pressed down, providing information about the pressed key, including modifiers. keypress is triggered specifically when a character key is pressed, providing information about the typed character without details on modifiers. Keydown fires continuously as long as the key is held down.

In all the above examples, we have used the addEventListener but these events can be listened to without this function as well. This is because of you can assign event handlers directly to specific properties. However, keep in mind that using addEventListener is generally considered a better practice because it allows you to attach multiple event handlers to the same event, and it separates JavaScript logic from the HTML structure.

Example: Without using addEventListener method

In this example, we have an input box. When it detects a keydown event (onkeydown), the handleKeyDown function is called and when it detects a keyup event (onkeyup) it calls the handleKeyUp function. Both the functions print appropriate messages to the screen.

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
   <div>Enter some text:
      <input onkeydown="handleKeyDown(event)" onkeyup="handleKeyUp(event)">
   </div>
   <div id="output"></div>
   <script>
      function handleKeyDown(event) {
         document.getElementById('output').innerHTML+=
         "Key pressed: " + event.key+'<br>Key code: ' + event.keyCode+'<br>';
      }
      function handleKeyUp(event) {
         document.getElementById('output').innerHTML+=
         "Key released: ' + event.key+'<br><br>";
```

```
        }
    </script>
</body>
</html>
```