

Reacting to Keyboard Events

This lesson is about the type of keyboard events that are occurred by using a keyboard.



The most common solution for reacting to key presses on a keyboard involves handling `keypress` events that happen on a web page (the DOM body `element`, which corresponds to the global variable called `document` in JavaScript).

The following example shows in the console the character associated to a pressed key. The character info is given by the `charCode` property of the `Event` object associated to the event. This property returns a numerical value (called *Unicode* value) that can be translated to a string value by the `String.fromCharCode()` method.

Output

JavaScript

```
1 // Show the pressed character
2 document.addEventListener("keyp
3     console.log(`You pressed the
4 });
```



Console Clear

To manage the press and release of any key (not only the ones producing characters), you'll use the `keydown` and `keyup` events. This example uses the same function to manage two events. This time, the key's code is accessible in the `keyCode` property of the `Event` object.

Output

JavaScript

```
1 // Show the pressed character
2 document.addEventListener("keydown", (e) => {
3   console.log(`You pressed the character: ${e.key}`);
4 });
5
6 // Show information on a keyboard event
7 const keyboardInfo = e => {
8   console.log(`Keyboard event: ${e.key}, ${e.keyCode}, ${e.code}`);
9 };
10
11 // Integrate this function into the keydown event listener
12 document.addEventListener("keydown", (e) => {
13   keyboardInfo(e);
14 });
```



Console

 Clear

This results demonstrates that the launch order of keyboard-related events is as follows: `keydown` -> `keypress` -> `keyup`.

The `keydown` is fired several times when a key is kept pressed.