## 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 <a href="https://charcode">charCode</a> property of the <a href="https://example.com/Event">Event</a> object associated to the event. This property returns a numerical value (called <a href="https://example.com/Unicode">Unicode</a> value) that can be translated to a string value by the <a href="https://example.com/String.FromCharCode">String.FromCharCode</a> method.

```
JavaScript

// Show the pressed character
document.addEventListener("keypress", e => {
  console.log(`You pressed the ${String.fromCharCode(e.charCode)} key`);
});
```



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

// Show the pressed character
document.addEventListener("keypress", e => {
   console.log(`You pressed the ${String.fromCharCode(e.charCode)} key`);
});

// Show information on a keyboard event
const keyboardInfo = e => {
   console.log(`Keyboard event: ${e.type}, key: ${e.keyCode}`);
};

// Integrate this function into key press and release:
document.addEventListener("keydown", keyboardInfo);
document.addEventListener("keyup", keyboardInfo);
```



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.