Cheatsheets / **Building Interactive JavaScript Websites**

DOM Events with JavaScript

javascript event

On a webpage, a trigger such as a user interaction or browser manipulation can cause a client-side JavaScript event to be created. Events can be used to maniuplate the DOM by executing a JavaScript function.

Events can include anything from a click to hovering a mouse over an element to a webpage loading or being refreshed. Events are defined as a part of the JavaScript API built into the web browser.

```
// An event is triggered when a user clicks on the #button element,
// which then sets the #button element's background-color to blue.
$('#button').on('click', event => {
  $(event.currentTarget).css('background-color', 'blue');
});
```

Event handler

When an event fires in JavaScript (such as a keystroke or mouse movement), an *event handler* runs in response. Each event handler is registered to an element, connecting the handler to both an element and a type of event (keystroke, eg.). A method called an *event listener* "listens" for an event to occur, specifies what should happen as a response, and calls the event handler.

JS Event Handlers

The goal of JavaScript is to make a page dynamic, which frequently means responding to certain events (for example, button clicks, user scrolls, etc). DOM elements can have

functions hook onto events. The functions are called *event handlers* and the DOM element is known as an *event target*.

The example code block shows how to *register* a function as an *event handler*. The property name for event handlers starts with 'on' with the event appended afterwards. Examples: `onload`, `onclick`, `onfocus`, `onscroll`.

```
//Assuming there is an element with ID='test' on the page

document.getElementById('test').onclick = function(e) {
  alert('Element clicked!');
};
```

Event object

Event handler functions are passed an argument called an *event object*, which holds information about the event that was fired.

Event objects store information about the event target, the event type, and associated listeners in properties and methods. For example, if we wanted to know which key was pressed, the event object would store that information.

.addEventListener()

The `.addEventListener()` method attaches an event handler to a specific event on an event target. The advantage of this is that you can add many events to the event target without overwriting existing events. Two arguments are passed to this method: an event name as a string, and the event handler function. Here is the syntax:

```
eventTarget.addEventListener("event", eventHandlerFunction);
```

.removeEventHandler()

We can tell our code to listen for an event to fire using the `.addEventListener()` method.

To tell the code to **stop** listening for that event to fire, we can use the `.removeEventListener()` method. This method takes the same two arguments that were passed to `.addEventListener()`, the event name as a string and the event handler function. See their similarities in syntax:

```
eventTarget.addEventListener("event", eventHandlerFunction);

eventTarget.removeEventListener("event", eventHandlerFunction);
```

Mouse events

A *mouse event* fires when a user interacts with the mouse, either by clicking or moving the mouse device.

- `click` events are fired when the user presses **and** releases a mouse button on an element.
- `mouseout` events are fired when the mouse leaves an element.
- `mouseover` events are fired when the mouse enters an element's content.
- `mousedown` events are fired when the user presses a mouse button.
- `mouseup` events are fired when the user releases the mouse button.

Keyboard events

Keyboard events describe a user interaction with the keyboard. Each event describes a separate interaction with a key on the keyboard by the user, which are then stored with the `.key` property.

- `keydown` events are fired when the key is first pressed.
- `keyup` events are fired when the key is released.
- `keypress` events are fired when the user presses a key that produces a character value (aka is not a modifier key such as CapsLock).

Related Courses

Course

Building Interactive JavaScript Websites

Learn the Document Object Model, the interface between JavaScrip