Quick summary of event management in JavaScript

HTML5 events

There is no input or output in JavaScript. We treat *events* caused by user actions as inputs, and we manipulate the DOM structure as output.

In any case, the events are called DOM events, and we use the DOM APIs to create *event handlers*.

How to listen to events

There are three ways to manage events in the DOM structure.

Method #1: declare an event handler in the HTML code

1. <div id="someDiv" **onclick**="alert('clicked!')"> content of the div </div>

Not the recommended way to handle events. It is *deprecated*. Mixing 'visual layer' (HTML) and 'logic layer' (JavaScript) in one place is really bad practice and causes a host of problems during development.

Method #2: attach an event handler to an HTML element in JavaScript

1. document.getElementById('someDiv').**onclick** = function() {

2.   alert('clicked!');

3. }

This method is fine, but you will not be able to attach multiple *listener* functions. If you need to do this, use the version shown below.

Method #3: register a callback to the event listener with the addEventListener method (preferred method)

1. document.getElementById('someDiv').**addEventListener**('click', function() {

2.   alert('clicked!');

3. }, false);

Note that the third parameter describes whether the *callback* has to be called during the captured phase. Just set it to false or ignore it (you can even pass only two parameters to the addEventListener function call and do not set this boolean parameter at all).

Details of the DOM event are passed to the event listener function

When you create an *event listener* and attach it to an element, the listener will create an ***event object*** to describe what happened. This object is provided as a parameter of the *callback function*:

1. element.addEventListener('click', function(**event**) {
2. **// now you can use event object inside the callback**
3. }, false);

Depending on the type of event you are listening to, you will consult different properties from the event object in order to obtain useful information such as: "which keys are pressed down?", "what is the location of the mouse cursor?", "which mouse button has been clicked?", etc.

In the following lessons, we will remind you how to deal with the keyboard and the mouse.

Reference tables for events and properties/methods

These tables are provided as a reference. They are a compilation of the most common event types sorted by domain (key, mouse, forms, etc.). For each domain you will see the most useful event types and their properties. In the following course pages, we will show examples that use most of the events displayed in these tables.

Event object

Most useful common properties:

type	Returns the name of the event
target	Returns the element that triggered the event

Most useful common methods:

preventDefault()	Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur
stopPropagation()	Prevents further propagation of an event during event flow

Page

Events related to the page lifecycle

There are many other events related to the page life cycle. Below are the most useful ones for an intro course:

load	This event occurs when an object has loaded (including all its resources: images, etc.) This event is very useful when you want to run JS code and be sure that the DOM is ready (in other words, be sure that a document.getElementById(...) or document.querySelector(...) will not raise an error because the document has not been loaded and elements you are looking for are not ready.
resize	The event occurs when the document view is resized. Usually we get the new size of the window inside the event listener using var w = window.innerWidth; and var h = window.innerHeight;
scroll	The event occurs when an element's scrollbar is being scrolled. Usually, in the scroll event listener, we use things such as: var max = document.body.scrollHeight - innerHeight; var percent = (pageYOffset / max); ...to know the percentage of the scroll in the page.

Page event properties

There are no particular properties that need to be mentioned here. Usually, the load event listener corresponds to a JavaScript function that can be seen as "the main" function of your Web Application. It is good practice to start everything after the page has been completely loaded. In the resize listener, you get the new size of the window, or the new size of some HTML elements in the page (as they might have been resized too when the window was resized) and then you do something (redraw a graphic in an HTML canvas that takes into account the new canvas size, for example).

Keyboard

Event types related to keyboard

keydown	The event occurs when the user presses a key
keyup	The event occurs when the user releases a key
keypress	The event occurs when the user presses presses a key down, where that key produces a character value

keyboardEvent properties

keyCode (now deprecated)	Returns the Unicode character code of the key that triggered the onkeypress ,onkeydown or onkeyup event
shiftKey	Returns whether the "shift" key was pressed when the key event was triggered
ctrlKey	Returns whether the "ctrl" key was pressed when the key event was triggered
altKey	Returns whether the "alt" key was pressed when the key event was triggered

Mouse

Event types related to mouse

click	The event occurs when the user clicks on an element (presses a button and releases it)
dblclick	The event occurs when the user double-clicks on an element
mousedown	The event occurs when a pointing device button is pressed while the pointer is inside the element
mouseup	The event occurs when a user releases a mouse button over an element

mousemove	The event occurs when the pointer is moving while it is over an element
mouseenter	The event occurs when the pointer is moved onto an element
mouseleave	The event occurs when the pointer is moved out of an element
mouseover	The event occurs when the pointer is moved onto an element, or onto one of its children
contextmenu	The event occurs when the user right-clicks on an element to open a context menu

MouseEvent properties

button	Returns which mouse button was pressed when the mouse event was triggered
clientX and clientY	Returns the coordinates of the mouse pointer, relative to the element coordinate system that triggered the event
pageX and pageY	Returns the coordinates of the mouse pointer, relative to the document, when the mouse event was triggered
screenX and screenY	Returns the coordinates of the mouse pointer, relative to the screen, when an event was triggered
altKey, ctrlKey, shiftKey	Returns whether the "alt, ctrl and shift" key was pressed when an event was triggered
detail	Returns a number that indicates how many times the mouse was clicked

Forms

Events related to forms

input	The event occurs when an element gets user input (e.g., a key is typed on an input field, a slider is moved, etc.)
change	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>). A change event listener on a slider will generate an event when the drag/move ended, while input events will be useful to do something as the slider is being moved.
focus	The event occurs when an element gets focus (e.g., the user clicks in an input field)
blur	The event occurs when an element loses focus (e.g., the user clicks on another element)
select	The event occurs after the user selects some text (for <input> and <textarea>)
submit	The event occurs when a form is submitted

FormEvent properties

There are no particular properties that need to be mentioned here. Usually, on a form event listener, we check the content of the different input fields, using their value property. See examples in the course, in the part dealing with form events.