BROWSER EVENTS

Boris Bocharov, Maria Voevodina,
Nataliia Braterska, Anastasiia Dashkovska

To respond to visitor actions and internal interaction of scripts, there are events.

An event is a signal from the browser that something has happened.

At the dawn of the World Wide Web, web developers had to deal with only a small number of events: "load", "click", "mouseover" and others. These fairly old types of events are well supported by all browsers [1-5]. As the web platform evolved, more powerful application interfaces were included, and the number of events increased significantly. There is no standard that defines a complete set of events, and the number of supported events continues to increase rapidly. These new events are defined in the following two sources:

Specification of "DOM Level 3 Events", which after years of stagnation has been actively developed under the auspices of the W3C consortium.

Many new application interfaces in the HTML5 specification (and related additional specifications) define new events used, for example, to manage the history of visits, the drag-and-drop mechanism (drag and drop), message exchange between documents, and audio and video playback videos.

Example of an event in an HTML tag.

```
<input name = "compute" type = "button" onClick = "myCompute ()"
value = "compute">
```

Event handler:

```
<script>
function myCompute(){
    var elem = document.getElementById('my');
    var cnt = new HTML_Container(elem);
    var myf = new FORM_data("myForm");
    var a = myf.get("a");
```

```
    var b = myf.get("b");
    var c = myf.get("c");
    var tri = new Triangle(a, b, c);
    cnt.out("<font color='#0000FF' size='+3'>tringle</font><br>");
//    cnt.add(tri.toString()+"<br>");
  cnt.add(tri+"<br>");
}
</script>
```

Document download events

Most web applications absolutely need a web browser to notify them when the document has finished loading and it will be ready to perform operations on it. The load event in the Window object serves this purpose. The load event is raised only after the document and all its images have been fully loaded. However, you can usually run scripts immediately after parsing the document before the images are loaded. You can significantly reduce the start time of the web application if you start executing scripts for events other than load.

The DOMContentLoaded event is fired as soon as the document is loaded, parsed by the parser, and all the pending scripts are executed. By this time, images and scripts with the async attribute can continue to load, but the document itself will already be ready for operations. This event was first introduced in Firefox and was subsequently borrowed by all other browser manufacturers, including Microsoft Corporation, which added support for this event in IE9. Despite the DOM prefix in the name, this event is not part of the standard of the DOM Level 3 Events event model, but it is standardized by the HTML5 specification.

Example of the load event.

```
window.onload = function(){
    var i,j,s;
    for(j=1; j<=13; j++){
      s="";
      if(j<10) s="0";
```

```
    s="Res"+s+j;
    ttvArrRadio[j-1] = document.getElementsByName(s);
    for (i = 0; i < ttvArrRadio[j-1].length; i++) {
      ttvArrRadio[j-1][i].addEventListener("change",ttvChange);
    }
  }
var btn = document.getElementById("res");
btn.addEventListener("click",ttvReset);}
```

Mouse events.

- click - occurs when the item is clicked on with the left mouse button
- contextmenu - occurs when you right-click on an item
- mouseover - occurs when an element is hovered over the mouse
- mousedown and mouseup - when the mouse button is pressed or pressed
- mousemove - when moving the mouse

Events on the controls.

- submit - the visitor sent the form <form>
- focus - the visitor focuses on the element, for example, clicks on <input>

Keyboard events:

- keydown - when the visitor presses the key
- keyup - when the visitor releases a key

Assigning Event Handlers

An event can be assigned a handler, that is, a function that will work as soon as an event has occurred.

It is thanks to handlers that JavaScript code can react to visitor actions.

There are several ways to assign an event handler. Now we will consider them, starting from the simplest.

Using the HTML attribute

The handler can be assigned directly in the markup, in an attribute called on <event>.

For example, to attach a click event to the input button, you can assign an onclick handler, like so:

```
<input value="Нажми меня" onclick="alert('Click!')" type="button">
```

When you click the button, the code specified in the onclick attribute will be executed.

Using the DOM Object Property

You can assign a handler using the DOM element's on <event> property.

Example of setting the click handler:

```
<input id="elem" type="button" value="Click me" />
<script>
  elem.onclick = function() {
    alert( 'Thank you' );
  };
</script>
```

If the handler is specified via an attribute, the browser reads HTML markup, creates a new function from the attribute content, and writes it to the onclick property.

The handler can also assign an existing function:

```
function sayThanks(){
  alert( 'Thak you!' ); }
elem.onclick = sayThanks;
```

Accessing an item through this

Inside the event handler, this refers to the current element, that is, to the one on which it worked.

You can use this to get properties or change an element.

In the code below, the button displays its content using this.innerHTML:

```
<button onclick = "alert (this.innerHTML)"> Click me </ button>
```

Common mistakes

If you are just starting to work with events - pay attention to the following features.

The function must be assigned as sayThanks, not sayThanks().

```
button.onclick = sayThanks;
```

If you add brackets, then sayThanks () will be the result of the function execution (and since there is no return in it, then onclick will get undefined). We also need a function.

But in HTML just the brackets are needed:

```
<input type="button" id="button" onclick="sayThanks()" />
```

This difference is easy to explain. When you create a handler with a browser from an attribute, it automatically creates a function from its contents. Therefore, the last example is actually the same as:

```
button.onclick = function() {
  sayThanks();
};
```

The DOM-property register has a value.

При назначении через DOM нужно использовать свойство onclick, а не ONCLICK.

Lack of assignment through the property

The fundamental drawback of the above methods of assigning a handler is the impossibility of hanging several handlers on one event.

For example, one part of the code wants to make it highlighted while the button is clicked, and the other part - to issue a message. You need to hang two handlers in different places.

In this case, the new handler will overwrite the previous one. For example, the following code actually assigns one handler, the last one:

```
input.onclick = function () {alert (1); } // ...
input.onclick = function () {alert (2);} // replace the previous
handler
```

The developers of the standards have understood this for a long time and proposed an alternative way of assigning handlers using special methods that are free from this shortcoming.

addEventListener and removeEventListener

The addEventListener and removeEventListener methods are a modern way to assign or remove a handler, and at the same time allow you to use as many arbitrary handlers as you like.

The handler is assigned by calling addEventListener with three arguments:

```
element.addEventListener(event, handler[, phase]);
```

event - The name of the event, for example click

handle - A reference to the function to be set by the handler.

phase - Optional argument, the "phase" on which the handler should trigger. This argument is rarely needed.

Removal of the handler is done by calling removeEventListener, the function must pass the same arguments that addEventListener had.

```
element.removeEventListener(event, handler[, phase]);
```

To delete, you need to pass exactly the function-handler that was assigned. This is how removeEventListener does not work:

```
elem.addEventListener( "click" , function() {alert('Thank you!')});
// ....
```

```
elem.removeEventListener( "click", function() {alert('Thank
you!')});
```

The removeEventListener is passed not the same function, but the other, with the same code, but it does not matter.

That's right:

```
function handler() {
  alert( 'Thank you!' );
}
input.addEventListener("click", handler);
// ....
input.removeEventListener("click", handler);
```

Pay attention - if the function is not saved anywhere, but simply passed to addEventListener, as in the previous code, then it will be impossible to get it back to remove the handler. There is no method that allows the event handlers assigned through addEventListener to be read.

Advantages of addEventListener

Some events can only be assigned via addEventListener.

The addEventListener method allows you to assign many handlers to one event.

Disadvantages of addEventListener.

The handler assigned via onclick is easier to remove or replace.

The onclick method is cross-browser.

Practical Work. DOM. Events

Change the HTML page index.html (see bellow) as follows.

Create a page index1.html (from index.html).

The amount in each column should be dynamically updated when a new switch value is selected.

The "Reset" button switches all the switches to "Not"
You can add JavaScript code only!. You can not change HTML tags!

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>John Smith</title>
<style>
.class_01 {
  background-color: #FFFFE0;
}
.class_02 {
  background-color: #FF0000;
}
.class_03 {
  background-color: #CCCCCC;
}
</style>
</head>
<body>
<form name="dialForm">
<table width="100%" border="1" cellspacing="1" cellpadding="5">
  <tr>
    <th width="20" scope="col" bgcolor="#FFFFE0">OK</th>
    <th width="20" scope="col" bgcolor="#FF0000">Err</th>
    <th width="20" scope="col" bgcolor="#CCCCCC">Not</th>
    <th width="50" scope="col">Pic</th>
    <th scope="col">Text</th>
  </tr>
<tr>
```

```
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res01" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res01" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res01" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im01.gif" /> </ td>
    <td valign = "top"> Multiple choice (closed question) with only
one correct answer (the student puts a mark in one of the circles)
    </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res02" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res02" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res02" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im01.gif" /> </ td>
    <td valign = "top"> Multiple choice (closed question) with one
or more correct answers (the student puts a mark in one or more
boxes)
  </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res03" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res03" value = "OK" /> </ td>
```

```
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res03" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im03.gif" /> </ td>
    <td valign = "top"> Alternative question (True / False) </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res04" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res04" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res04" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im04.gif" /> </ td>
    <td valign = "top"> Numeric question </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res05" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res05" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res05" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im05.gif" /> </ td>
    <td valign = "top"> Calculated question </ td>
  </ tr>
<tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res06" value = "OK" /> </ td>
```

```
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res06" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res06" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im06.gif" /> </ td>
    <td valign = "top"> Nested Answers </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res07" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res07" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res07" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im07.gif" /> </ td>
    <td valign = "top"> The question of correspondence </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res08" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res08" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res08" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im08.gif" /> </ td>
    <td valign = "top"> Short answer (open question) </ td>
  </ tr>
  <tr>
```

```
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res09" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res09" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res09" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im09.gif" /> </ td>
    <td valign = "top"> Description </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res10" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res10" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res10" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im10.gif" /> </ td>
    <td valign = "top"> Essays </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res11" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res11" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res11" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im11.gif" /> </ td>
    <td valign = "top"> The goal in the image </ td>
  </ tr>
```

```
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res12" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res12" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res12" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im12.gif" /> </ td>
    <td valign = "top"> All or Nothing </ td>
  </ tr>
  <tr>
    <td align = "center" valign = "top" class = "class_01"> <input
type = "radio" name = "Res13" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_02"> <input
type = "radio" name = "Res13" value = "OK" /> </ td>
    <td align = "center" valign = "top" class = "class_03"> <input
type = "radio" name = "Res13" value = "OK" checked /> </ td>
    <td align = "center" valign = "top"> <img src = "ttv_img /
im13.gif" /> </ td>
    <td valign = "top"> On the correspondence (displacement) </ td>
  </ tr>
  <tr>
    <td align="center" valign="top" bgcolor="#FFFFE0"><b><div
id="totOK">0</div></b></td>
    <td align="center" valign="top" bgcolor="#FF0000"><b><div
id="totERR">0</div></b></td>
    <td align="center" valign="top" bgcolor="#CCCCCC"><b><div
id="totNOT">13</div></b></td>
    <td align="center" valign="top"> </td>
    <td valign="top"><input name="Res" type="button"
value="Reset"/></td>
```

```
  </tr>
</table>
</form>
</body>
</html>
```

References:

1. Бочаров Б.П. Інформаційні технології в освіті : монографія / Б.П. Бочаров, М.Ю. Воєводіна; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків: ХНУМГ ім. О. М. Бекетова, 2015. – 197 с.

2. Bocharov Boris. AUTOMATIZED WEB PAGES PARSING AND CREATION / Boris Bocharov, Maria Voevodina // Information technologies in education: electronic supplement to the journal "Educational Institutions Libraries". – 2017. – N5, p. 1-5.

3. Bocharov Boris. Basic Concepts of the JavaScript / Boris Bocharov, Maria Voevodina, Nataliia Braterska, Anastasiia Dashkovska // Information technologies in education: electronic supplement to the journal "Educational Institutions Libraries". – 2018. – N7, p. 1-48.

4. Bocharov Boris. OBJECT ORIENTED PROGRAMMING IN THE JAVASCRIPT / Boris Bocharov, Maria Voevodina, Nataliia Braterska, Anastasiia Dashkovska // Information technologies in education: electronic supplement to the journal "Educational Institutions Libraries". – 2018. – N8, p. 1-11.

5. Bocharov Boris. OBJECT MODELS: DOM, BOM AND JS / Boris Bocharov, Maria Voevodina, Nataliia Braterska, Anastasiia Dashkovska // Information technologies in education: electronic supplement to the journal "Educational Institutions Libraries". – 2018. – N8, p. 30-55.