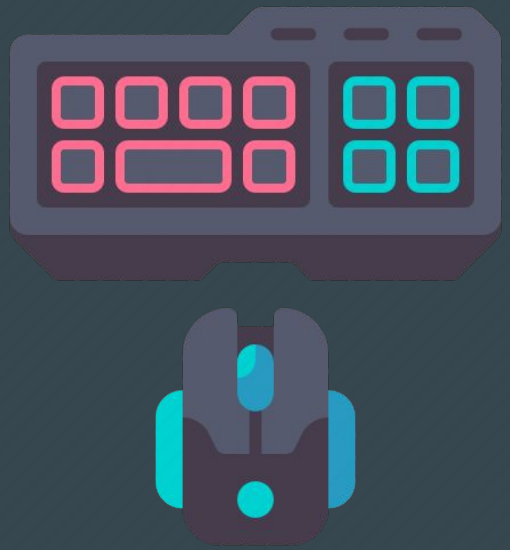
M03 - Document Object Model

•••

Events

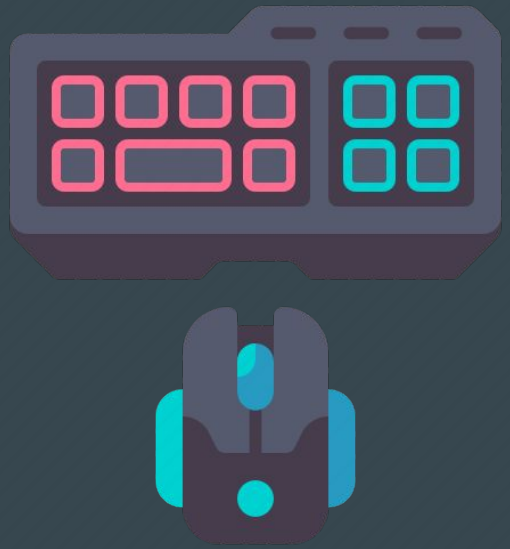
Events

1. Introduction to Events
2. Type of Events
3. Forms



Events

1. Introduction to Events
2. Type of Events
3. Forms



Events

1. Introduction to Events

- We can associate JavaScript code when an event occurs
- Types of events:
  - system (e.g. loading an HTML page)
  - user (e.g. click a button)
- To react to events, a handler is assigned - a function performed when a event occurs
- Ways to associate handlers:
  - HTML attribute
  - DOM property
  - addEventListener method

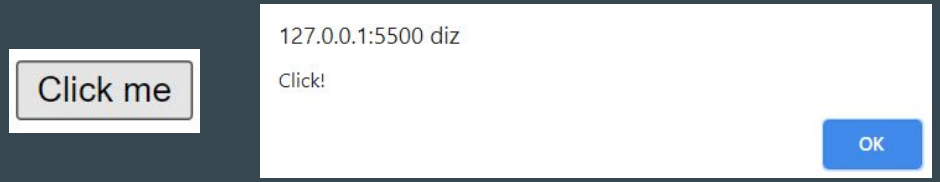
Events

1. Introduction to Events

- Ways to associate handlers:
  - HTML attribute
  - DOM property
  - addEventListener method

- A handler can be defined in HTML with an attribute on<event>
- For example, to assign a click handler to a button, we can use the onclick:

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



Events

1. Introduction to Events

- Ways to associate handlers:
  - HTML attribute
  - DOM property
  - addEventListener method

```
<p onclick='hello(this)'>Hello World!</p>

<script>
    function hello(obj) {
        alert(obj.innerHTML);
    }
</script>
```

- Use of the reserved word this
  - creates a reference to the element itself
  - in the next example, this creates a reference to the element <p>

Events

1. Introduction to Events

- Ways to associate handlers:
  - HTML attribute
  - DOM property
  - addEventListener method

- A handler can be defined as a DOM property called on<event>
- Similar technique to previous, but preferable because it separates HTML from JS

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

Events

1. Introduction to Events

- Method AddEventListener attaches an event handler to an element without overwriting existing event handlers
- Syntax: element.addEventListener(event, handler[, options])
- You can add event handlers
  - to an element
  - of the same type to an element, that is for instance, two click events
  - to any DOM object, not just HTML elements (e.g. window object)
- JavaScript is separate from HTML markup
  - better readability
  - allows you to add event handlers even if you don't control the HTML

Events

1. Introduction to Events

- Syntax: element.addEventListener(event, handler[, options])
  - event
    - Name of the event, e.g. click
  - handler
    - The handler function
  - options
    - An additional optional object with the following properties:
      - once: if true, the listener is automatically removed after being triggered
      - capture: allows you to control the propagation of events (bubbling - false, capturing - true)
      - passive: if true, the handler will not do preventDefault(), we will address this later

Events

1. Introduction to Events

- Syntax: element.addEventListener(event, handler[, options])

```
<p id='p1'>Hello World!</p>
<script>
    document.getElementById('p1').addEventListener('click', myFunction);

    function myFunction() {
        alert('Hello World!');
    }
</script>
```

With named function

With anonymous function

```
<p id='p1'>Hello World!</p>
<script>
    document.getElementById('p1').addEventListener('click', function () {
        alert('Hello World!');
    });
</script>
```

Events

1. Introduction to Events
- Handler removal
  - Syntax: element.removeEventListener(event, handler)
  - Example:

```
<p id='p1'>Hello World!</p>

<script>
    function handler() {
        alert('Thank you!');
    }

    const myP = document.getElementById('p1');
    myP.addEventListener('click', handler);
    myP.removeEventListener('click', handler);
</script>
```

Events

1. Introduction to Events

- Never reference DOM elements before they are in the DOM tree!
- This example:

```
<script>
    alert(document.getElementById('p1').innerHTML);
</script>
<p id='p1'>Hello World!</p>
```

- Will throw this error in the console, since the script is loaded before the paragraph is in the DOM tree:

```
Uncaught TypeError: Cannot read property 'innerHTML' of null
    at index.html:14
```

Events

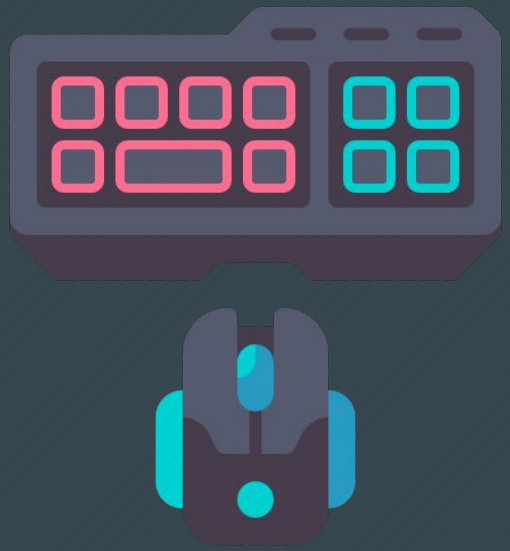
1. Introduction to Events

- Two alternatives:
  - Put all the script in the end of the tag <body>
  - Assign a handler to the load event which execution will occur after the creation of the DOM tree in memory

```
<script>
    window.addEventListener('load', function () {
        alert(document.getElementById('p1').innerHTML);
    });
</script>
<p id='p1'>Hello World!</p>
```

Events

1. Introduction to Events
2. Type of Events
3. Forms



Events

2. Type of Events

- Window
- Mouse
- Keyboard
- Widgets

Events

2. Type of Events

- Window
  - load - fully available window
  - unload - end of window availability
  - resize - changing dimensions
  - scroll - window scroll navigation

```
window.addEventListener('load', function () {
    console.log('The DOM tree is fully loaded!');
});
```

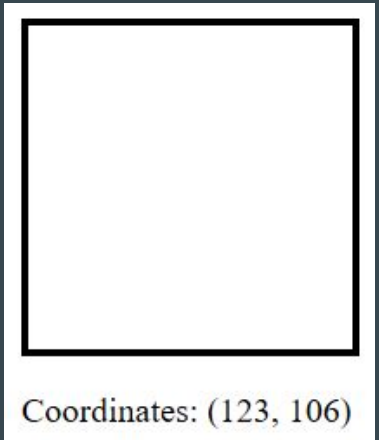
Events

2. Type of Events

- Mouse
  - click - start and end of pressure on a button
  - doubleclick - two clicks in quick succession
  - mousedown - start of pressing on a button
  - mouseup - end of pressing on a button
  - mousemove - mouse movement

```
<div id="div" style="width: 150px; height: 150px; border: solid;"></div>
<p id="result"></p>
<script>
    const myDiv = document.getElementById('div');
    myDiv.addEventListener('mousemove', function (event) {
        const coor = `Coordinates: (${event.clientX}, ${event.clientY})`;
        document.getElementById('result').innerHTML = coor;
    });
</script>
```

event object



Events

2. Type of Events

- Keyboard
  - keypress - key pressed
  - keydown - start of key press
  - keyup - end of key press

```
<input type='text' id='demo'></input>
<script>
    const myInput = document.getElementById('demo');
    myInput.addEventListener('keypress', function (event) {
        console.log(`You pressed the key ${event.key}`);
    });
</script>
```

event object

key pressed

Events

2. Type of Events

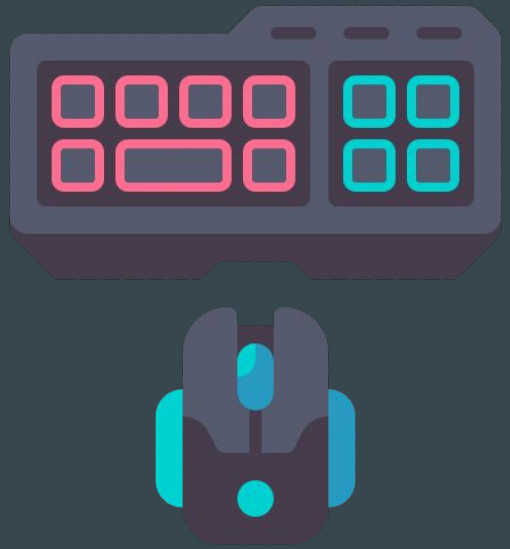
- Widgets
  - change - changing values (input, select and textarea)
  - focus - receive focus (input and textarea)
  - select - text selection (input and textarea)
  - submit - form submission (form)

```
<input type='text' id='demo'></input>
<script>
    const myInput = document.getElementById('demo');
    myInput.addEventListener('change', function () {
        console.log(`You changed the text to: ${myInput.value}`);
    });
</script>
```

new input value

Events

1. Introduction to Events
2. Type of Events
3. Forms

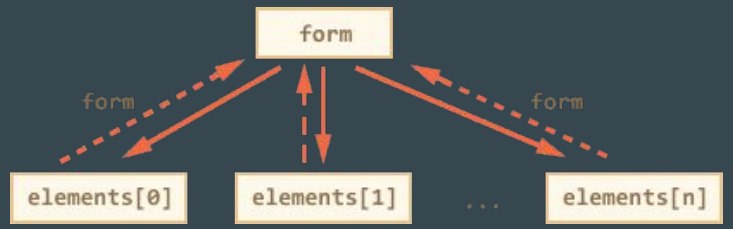


Events

3. Forms

- Forms and their elements have many special properties and events
- Forms in an HTML document are members of a collection document.forms
- To access the elements we can use the collection elements
- Access can be done by name or index

```
<form>
    <input name='one' value='1'>
    <input name='two' value='2'>
</form>
<script>
    // get the form
    const form = document.forms.my; // element <form name="my">
    // get the input with name "one"
    const elem = form.elements.one; // element <input name="one">
    alert(elem.value); // 1
</script>
```

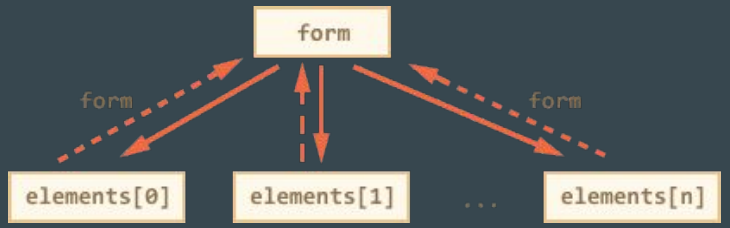


Events

3. Forms

- Note that there may be several elements with the same name (e.g. radio buttons)
- In this case, form.elements[name] is a collection, for example:

```
<form name='my'>
    <input type='radio' name='age' value='10'>
    <input type='radio' name='age' value='20'>
</form>
<script>
    const form = document.forms[0];
    const ageElems = form.elements.age;
    alert(ageElems[0].value); // 10, the first input value
</script>
```



Events

3. Forms

- Main form elements:
  - Input
  - Textarea
  - Select and option

Events

3. Forms

- Input:
  - Using the value property to set/get the text box value

```
<form>
    <input id='input1' value='12'>
</form>

<script>
    const myInput = document.getElementById('input1');
    console.log(myInput.value); // get the current value (12)
    myInput.value = 15; // define a new value to the text input
    console.log(myInput.value); // get the current value (15)
</script>
```

propriedade value

```
<form>
    <input id='input1' type='checkbox'>
</form>

<script>
    const myCheckbox = document.getElementById('input1');
    console.log(myCheckbox.checked); // get the current value (false)
    myCheckbox.checked = true; // activate the checkbox
    console.log(myCheckbox.checked); // get the current value (true)
</script>
```

propriedade checked

  - Use of the checked property to enable/disable the checkbox

Events

3. Forms

- Input
  - Main events:
    - change - triggered after loss of focus of the text box
    - input - triggered at every data input

```
<input type='text' id='input1'>
oninput: <span id='result'></span>
<script>
    const myInput = document.getElementById('input1');
    const myResult = document.getElementById('result');
    myInput.addEventListener('input', function () {
        myResult.innerHTML = myInput.value
    });
</script>
```

esmad × oninput: esmad

Events

3. Forms

- Select and option
  - An <select> element has 3 important properties:
    - select.options - the <option> element collection
    - select.value - the value of the chosen option
    - select.selectedIndex - the number (index) of the selected option

Events

3. Forms

- Select and option
  - Example of setting a value for an option of element <select>

- Three ways to set the value of a <select>:
  - Find the <option> and set option.selected to true
  - Set select.value for the value
  - Set select.selectedIndex to the option number

```
<select id='select'>
    <option value='apple'>Apple</option>
    <option value='peer'>Pear</option>
    <option value='banana'>Banana</option>
</select>

<script>
    const mySelect = document.getElementById('select');

    // the next 3 lines do the exactly same thing
    mySelect.options[2].selected = true;
    mySelect.selectedIndex = 2;
    mySelect.value = 'banana';
</script>
```

Events

3. Forms

- Select and option
  - Creating a new option
  - Syntax: option = new Option(text, value, defaultSelected, selected)
  - Parameters:
    - text - the text inside the option
    - value - the option value
    - defaultSelected - if true, then the selected attribute is created
    - selected - if true, the option is selected

```
<script>
    const option = new Option('Text', 'value');
    // creates <option value="value">Text</option>
</script>
```

Events

3. Forms

- Select and option
  - Example: fill an <select> element with data from an array

```
<select id='select'></select>

<script>
    const names = ['Mary', 'Peter', 'John'];

    const mySelect = document.getElementById('select');
    let result = '';
    for (const name of names) {
        result += `<option value='${name}'>${name}</option>`;
    }
    mySelect.innerHTML = result;
</script>
```

for...of loop
to iterate over array of strings

Use of template strings for composing <option> elements

Property innerHTML

Events

3. Forms

- Submission
  - The submit event is triggered when the form is submitted
  - Used to validate the form before sending it to the server or to abort the submission and process it in JavaScript

```
<form id='form1'>
    Name: <input type='text' required><br>
    Age: <input type='number' min='0' max='120' required><br>
    Driving License: <input type='checkbox'><br>
    <input type='submit' value='Register'>
</form>

<script>
    const myForm = document.getElementById('form1');
    myForm.addEventListener('submit', function () {
        // validation code
    });
</script>
```

HTML5 validations

JS validations

submit event

Events

3. Forms

- Submission
  - Use of event.preventDefault() to prevent the predefined action of the event
  - In this case your call prevents the form from being submitted to the server
  - Passing the event object handler as a parameter

```html
<form id='form1'>
    Name: <input type='text' required><br>
    Age: <input type='number' min='0' max='120' required><br>
    Driving License: <input type='checkbox'><br>
    <input type='submit' value='Register'>
</form>
```

Name:
Age:
Driving License:
Register

```javascript
const myForm = document.getElementById('form1');
myForm.addEventListener('submit', function (event) {
    // validate
    if (myForm.age.value < 18 && myForm.drivingLicense.checked === true) {
        alert('You cannot have a driving license with that age!');
        event.preventDefault();
    } else {
        alert('All Ok, the form will be submitted to the server!');
    }
});
```

event object

preventDefault() method