ΠΕΡΙΕΧΟΜΕΝΑ:

Events Ποντικιού: Click

- Τα παρακάτω events ενεργοποιούνται με κλικ στο ποντίκι:

event	Επεξήγηση
click	Κλικ σε στοιχείο
dblclick	Διπλό κλικ σε στοιχείο
mousedown	Όταν το κουμπί πατιέται
mouseup	Όταν το κουμπί απελευθερώνεται
auxclick	Για άλλα κουμπιά του moyse

Υπενθύμιση:

- Το event έχει το property button με τιμές:

Τιμή	Επεξήγηση
0	Αριστερό κουμπί
1	Μεσαίο κουμπί
2	Δεξί κουμπί
3-...	Άλλα κουμπιά του mouse

Σημείωση:

- Το click event ενεργοποιείται μόνο στην περίπτωση του αριστερού κλικ.

Παράδειγμα 1: event-mouse.html

```html
<div class="container">
  <button class="btn">Click Me</button>
</div>
```

```js
let button = document.querySelector("button")

button.addEventListener("click", (event)=>{
   console.log("click: " + event.button);
});
button.addEventListener("dblclick", (event)=>{
   console.log("double click: " + event.button);
});
button.addEventListener("mousedown", (event)=>{
   console.log("mouse down: " + event.button);
});
button.addEventListener("mouseup", (event)=>{
   console.log("mouse up: " + event.button);
});
```

```
mouse down: 0
mouse up: 0
click: 0
mouse down: 1
mouse up: 1
mouse down: 2
mouse up: 2
```

Events Ποντικιού: Κίνηση

- Τα παρακάτω events ενεργοποιούνται με κίνηση του ποντικίου:

event	Επεξήγηση
mousemove	Κίνηση του ποντικιού
mouseover	Το ποντίκι εισέρχεται σε στοιχείο
mouseout	Το ποντίκι εξέρχεται από στοιχείο
mouseenter	Το ποντίκι εισέρχεται στο όριο ενός στοιχείου
mouseleave	Το ποντίκι εξέρχεται από το όριο ενός στοιχείου

Παρατήρηση:

- Τα events mouseenter/mouseleave δεν περνάνε από φάση bubbling.
- Τα events mouseover/mouseout ενεργοποιούνται και όταν το ποντίκι βρεθεί πάνω από παιδί του τρέχοντος στοιχείου
  - (ενώ τα mouseenter/mouseleave δεν επηρεάζονται από τα παιδιά)

Παράδειγμα 2: event-mouse-move.html

```html
<div class="container">
  <button class="btn">Click Me <span>here!</span></button>
</div>
```

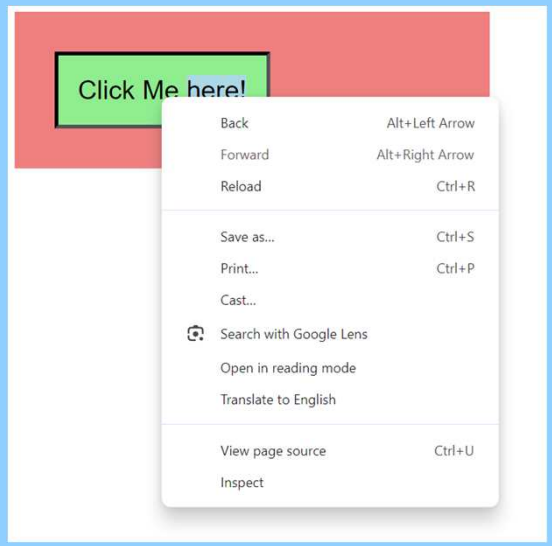
```js
let button = document.querySelector("button")

button.addEventListener("mousemove", (event)=>{
   console.log("move: x: " + event.offsetX + ", y: " + event.offsetY);
});
button.addEventListener("mouseenter", (event)=>{
   console.log("mouse enter x: " + event.offsetX + ", y: " + event.offsetY);
});
button.addEventListener("mouseover", (event)=>{
   console.log("mouse over x: " + event.offsetX + ", y: " + event.offsetY);
});
button.addEventListener("mouseleave", (event)=>{
   console.log("mouse leave x: " + event.offsetX + ", y: " + event.offsetY);
});
button.addEventListener("mouseout", (event)=>{
   console.log("mouse out x: " + event.offsetX + ", y: " + event.offsetY);
});
```

```
mouse over x: 10, y: 37
mouse enter x: 10, y: 37
move: x: 10, y: 37
move: x: 10, y: 36
move: x: 11, y: 35
move: x: 12, y: 35
move: x: 12, y: 34
move: x: 13, y: 35
move: x: 14, y: 36
move: x: 15, y: 37
mouse out x: 15, y: 38
mouse leave x: 15, y: 38
```

Events Ποντικιού: contextmenu

- Το contextmenu είναι ένα event, το οποίο ενεργοποιείται όταν ο χρήστης (συνήθως με δεξί κλικ) ανοίγει το «context menu» (το μενού που έχει επιλογές όπως save, back, reload κ.λπ.)



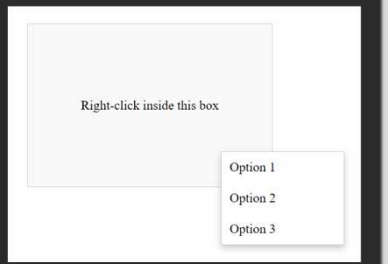
Παρατήρηση:

- Με χρήση του event, μπορούμε:
  - Να αποτρέψουμε την default συμπεριφορά
  - Να εμφανίσουμε δικό μας, custom menu
  - κ.λπ.

Παράδειγμα 3: event-mouse-context-menu.html

```html
<div id="rightClickArea">Right-click inside this box</div>
<ul id="customMenu">
  <li id="menuItem1">Option 1</li>
  <li id="menuItem2">Option 2</li>
  <li id="menuItem3">Option 3</li>
</ul>
```

```javascript
// Get references to the right-click area and custom menu
const rightClickArea = document.getElementById('rightClickArea');
const customMenu = document.getElementById('customMenu');
// Function to show the custom menu at the mouse pointer position
rightClickArea.addEventListener('contextmenu', (event) => {
  event.preventDefault(); // Prevent the default context menu
  customMenu.style.display = 'block';
  customMenu.style.left = `${event.pageX}px`;
  customMenu.style.top = `${event.pageY}px`;
});
// Hide the custom menu when clicking anywhere else
document.addEventListener('click', () => {
  customMenu.style.display = 'none';
});
// Add event listeners for the custom menu items
document.getElementById('menuItem1').addEventListener('click', () => {
  alert('Option 1 selected!');
  customMenu.style.display = 'none'; // Hide the menu after clicking
});
document.getElementById('menuItem2').addEventListener('click', () => {
  alert('Option 2 selected!');
  customMenu.style.display = 'none'; // Hide the menu after clicking
});
document.getElementById('menuItem3').addEventListener('click', () => {
  alert('Option 3 selected!');
  customMenu.style.display = 'none'; // Hide the menu after clicking
});
```



Events Ποντικιού: wheel

- Το παρακάτω event αφορά το scrolling με το μεσαιο κουμπί του ποντικιού:

event	Επεξήγηση
scroll	Ο χρήστης κάνει scroll

- Το event wheel ενημερώνει τα properties του αντικειμένου event

property	Επεξήγηση
deltaX	Αλλαγή στον x-άξονα
deltaY	Αλλαγή στον y-άξονα (<0: προς τα πάνω, >0 προς τα κάτω)
deltaMode	(0: pixels, 1: lines, 2: pages)

Παρατήρηση:

- Σύνηθες είναι, εφόσον θέλουμε να αλλάξουμε τη συμπεριφορά του μεσαίου κουμπιού, να συνδυάζεται με την event.preventDefault()
- Η αλλαγή κατά τον x-άξονα δεν εμπίπτει στο ποντίκι, αλλά σε άλλες συσκευές (όπως π.χ. touchpad)

Παράδειγμα 4: event-mouse-wheel.html

```html
<div class="scroll-container" id="scrollContainer">
  <p>Scroll the mouse wheel up or down to change the background color.</p>
</div>
```

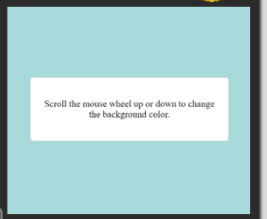
```js
const scrollContainer = document.getElementById('scrollContainer');

// Function to handle the wheel event
function handleWheelEvent(event) {
  // Prevent the default scroll behavior
  event.preventDefault();

  // Change the background color based on the scroll direction
  if (event.deltaY < 0) {
    // Scrolled up
    document.body.style.backgroundColor = '#a8dadc'; // Light blue
  } else if (event.deltaY > 0) {
    // Scrolled down
    document.body.style.backgroundColor = '#f4a261'; // Light orange
  }

  // Log the delta values for debugging
  console.log(`deltaY: ${event.deltaY}, deltaX: ${event.deltaX}, deltaMode: ${event.deltaMode}`);
}

// Add the wheel event listener to the scroll container
scrollContainer.addEventListener('wheel', handleWheelEvent);
```



Events Πληκτρολογίου

- Τα events πληκτρολογίου είναι:

event	Επεξήγηση
keydown	Πάτημα πλήκτρου
keyup	Απελευθέρωση πλήκτρου

Παρατήρηση:

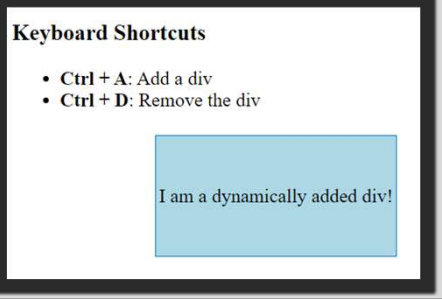
- Όπως είδαμε στο προηγούμενο μάθημα, τα events πληκτρολογίου, συνδυάζονται με τα properties του event:

Property	Επεξήγηση
key	Η τιμή του κουμπιού που πατήθηκε
code	Η φυσική τιμή του κουμπιού που πατήθηκε
altKey ctrlKey shiftKey metaKey	Αν έχουν πατηθεί τα τροποποιητικά κουμπιά

Παράδειγμα 5: event-keyboard.html

```html
<div id="rightClickArea">Right-click inside this box</div>
<ul id="customMenu">
  <li id="menuItem1">Option 1</li>
  <li id="menuItem2">Option 2</li>
  <li id="menuItem3">Option 3</li>
</ul>
```

```js
function addDiv() {
    if (!document.getElementById('dynamicDiv')) {
        const newDiv = document.createElement('div');
        newDiv.id = 'dynamicDiv';
        newDiv.textContent = 'I am a dynamically added div!';
        document.body.appendChild(newDiv);
    }
}
function removeDiv() {
    const existingDiv = document.getElementById('dynamicDiv');
    if (existingDiv) {
        existingDiv.remove();
    }
}
document.addEventListener('keydown', (event) => {
    if (event.ctrlKey) { // Check if the Ctrl key is pressed
        switch (event.key.toLowerCase()) {
            case 'a': // Ctrl + A to add a div
                event.preventDefault();
                addDiv();
                break;
            case 'd': // Ctrl + D to remove the div
                event.preventDefault();
                removeDiv();
                break;
            default:
                break;
        }
    }
});
```



Events για Φόρμες: Focus και Blur

- Με τα ακόλουθα events ασχολούμαστε με το focus σε στοιχεία:

event	Επεξήγηση
focus	Το στοιχείο κερδίζει το focus
blur	Το στοιχείο χάνει το focus
focusin	Το στοιχείο κερδίζει το focus (χωρίς bubbling)
focusout	Το στοιχείο χάνει το focus (χωρίς bubbling)

Υπενθύμιση από μάθημα 2.2:

- Χρήσιμα είναι τα properties του αντικειμένου event: target και relatedTarget

Παρατήρηση:

- Συνήθως:
  - Με το event focus, κάνουμε highlight το στοιχείο που κέρδισε το focus.
  - Με το event blur, κάνουμε έλεγχο ότι η είσοδος του χρήστη είναι έγκυρη.

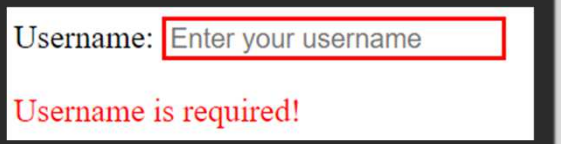
Παράδειγμα 6: event-form-focus-blur.html

```html
<form id="userForm">
  <label for="username">Username:</label>
  <input type="text" id="username" placeholder="Enter your username">
  <p id="errorMsg" style="color: red; display: none;">Username is required!</p>
</form>
```

```js
const usernameInput = document.getElementById('username');
const errorMsg = document.getElementById('errorMsg');

// Add highlight class when the input gains focus
usernameInput.addEventListener('focus', () => {
  usernameInput.classList.add('highlight');
});

// Remove highlight and validate input when it loses focus
usernameInput.addEventListener('blur', () => {
  usernameInput.classList.remove('highlight');
  if (usernameInput.value.trim() === '') {
    errorMsg.style.display = 'block';
    usernameInput.classList.add('error');
  } else {
    errorMsg.style.display = 'none';
    usernameInput.classList.remove('error');
  }
});
```



Events για Φόρμες: Change και Input

- Με τα ακόλουθα events ασχολούμαστε με αλλαγές σε στοιχεία της φόρμας:

event	Επεξήγηση
change	Ενεργοποιείται μετά την αλλαγή στο στοιχείο, αμέσως μετά την απώλεια του focus
input	Ενεργοποιείται σε πραγματικό χρόνο όταν γίνεται αλλαγή στο στοιχείο
select	Ενεργοποιείται όταν ο χρήστης επιλέγει κείμενο

Σημείωση:

- Αυτά τα events δεν έχουν κοινή συμπεριφορά για όλα τα controls:
  - Το event input είναι κυρίως για controls εισαγωγής χαρακτήρων (π.χ. textbox, input[number], input[date] κ.λπ.)
    - Συνήθως χρησιμοποιείται για validation πεδίων σε πραγματικό χρόνο
  - Ενώ το event change μπορεί να εντοπίσει τις αλλαγές σχεδόν σε όλα τα controls (π.χ. checkbox, combo box, select κ.λπ.)
    - Συνήθως χρησιμοποιείται για validation σε αυτά τα controls

Παράδειγμα 7: event-form-input-change.html

HTML

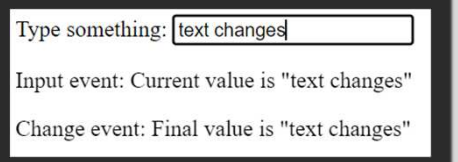
```html
<label for="textInput">Type something:</label>
<input type="text" id="textInput" placeholder="Type here...">
<p id="inputEventOutput">Input event: No changes yet.</p>
<p id="changeEventOutput">Change event: No changes yet.</p>
```

JS

```js
const textInput = document.getElementById('textInput');
const inputEventOutput =
document.getElementById('inputEventOutput');
const changeEventOutput =
document.getElementById('changeEventOutput');

// `input` event: Fires immediately as the value changes
textInput.addEventListener('input', () => {
  inputEventOutput.textContent = `Input event: Current value is
"${textInput.value}"`;
});

// `change` event: Fires only when the input loses focus after a change
textInput.addEventListener('change', () => {
  changeEventOutput.textContent = `Change event: Final value is
"${textInput.value}"`;
});
```



Events για Φόρμες: Submit και Reset

- Με τα ακόλουθα events ασχολούμαστε με την υποβολή και τον καθαρισμό της φόρμας:

event	Επεξήγηση
submit	Όταν το κουμπί submit πατιέται (ή όταν πατιέται Enter)
reset	Όταν το κουμπί reset πατιέται

Σημείωση:

- Το submit μπορεί να χρησιμοποιηθεί για validation ή ακόμη και να αποστείλει τα δεδομένα μέσω JS και όχι με τον default μηχανισμό.
- Χρήσιμη μπορεί να φανεί η μέθοδος preventDefault() του αντικειμένου event, για να αποτρέψουμε την default λειτουργία.

Παράδειγμα 8: event-form-submit-reset.html

```html
<form id="simpleForm">
  <label for="inputField">Enter something:</label>
  <input type="text" id="inputField" placeholder="Type something..."><br><br>
  <button type="submit">Submit</button>
  <button type="reset">Reset</button>
</form>
<p id="message"></p>
```

```js
const simpleForm = document.getElementById('simpleForm');
const inputField = document.getElementById('inputField');
const message = document.getElementById('message');

// Handle form submission
simpleForm.addEventListener('submit', (event) => {
    event.preventDefault(); // Prevent the default form submission behavior

    // Simple validation to check if the input is empty
    if (inputField.value.trim() === '') {
        message.textContent = 'The field cannot be empty!';
        message.className = 'error';
    } else {
        message.textContent = 'Form submitted successfully!';
        message.className = 'success';
        // Here you would normally handle the form submission, e.g., send data to a server
    }
});

// Handle form reset
simpleForm.addEventListener('reset', (event) => {
    // Confirm reset action
    const confirmReset = confirm('Are you sure you want to reset the form?');
    if (!confirmReset) {
        event.preventDefault(); // Prevent the form from resetting
    } else {
        message.textContent = ''; // Clear any messages when the form is reset
    }
});
```