ΠΕΡΙΕΧΟΜΕΝΑ:

ΜΑΘΗΜΑ 1.8: ELEMENT: SCROLLING

1.1. scrollTop

Browser psounis

Scrolling:

- Διακρίνουμε δύο περιπτώσεις:
  - Scrolling στο παράθυρο (window)
  - Scrolling σε συγκεκριμένο στοιχείο (element)

Το property scrollTop:

- **Σε στοιχείο** (συνήθως κείμενο με overflow: auto ή overflow: scroll και συγκεκριμένο ύψος):

Property	Επεξήγηση
scrollTop	read: Πλήθος Pixels που έχει γίνει scroll κάθετα set: Θέτοντας τιμή (σε px) γίνεται scroll σε αυτό το ύψος

- **Σε παράθυρο**(Αντίστοιχη ιδιότητα στην οποία έχουμε πρόσβαση με δύο τρόπους):

Property	Επεξήγηση
scrollY (του window)	read: Πλήθος Pixels που έχει γίνει scroll κάθετα
document.documentElement.scrollTop	Η ιδιότητα αυτή είναι χρήσιμη για παρωχημένους browsers (προτιμούμε την ιδιότητα scrollY)

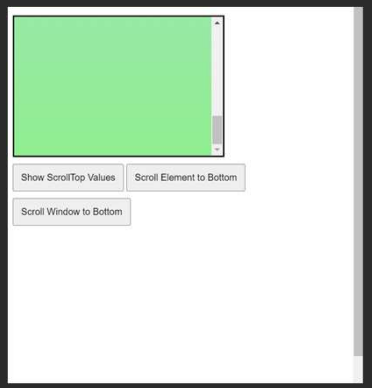
Σημείωση:

- Στο παράδειγμα για να κάνουμε scroll(), χρησιμοποιείται η scrollTo() που θα μελετήσουμε σε επόμενη διαφάνεια.

Παράδειγμα 1: scrollTop.html

```html
<div class="scrollable-container" id="scrollableContainer">
  <div class="content">
    <p>Scroll inside this container.</p>
  </div>
</div>
<button id="showScrollTopBtn">Show ScrollTop Values</button>
<button id="scrollElementBtn">Scroll Element to Bottom</button>
<button id="scrollWindowBtn">Scroll Window to Bottom</button>
```

```js
// Get the button elements
const showScrollTopBtn = document.getElementById('showScrollTopBtn');
const scrollElementBtn = document.getElementById('scrollElementBtn');
const scrollWindowBtn = document.getElementById('scrollWindowBtn');
const container = document.getElementById('scrollableContainer');

// Event listener to show the scrollTop values for both the window and the element
showScrollTopBtn.addEventListener('click', () => {
  const windowScrollTop = window.scrollY || document.documentElement.scrollTop;
  const elementScrollTop = container.scrollTop;
  console.log(`Window scrollTop: ${windowScrollTop}`);
  console.log(`Element scrollTop: ${elementScrollTop}`);
});
// Event listener to scroll the container to the bottom
scrollElementBtn.addEventListener('click', () => {
  container.scrollTop = container.scrollHeight;
  console.log('Scrolled element to bottom');
});
// Event listener to scroll the window to the bottom
scrollWindowBtn.addEventListener('click', () => {
  window.scrollTo({
    top: document.body.scrollHeight,
    behavior: 'smooth'
  });
  console.log('Scrolled window to bottom');
});
```



ΜΑΘΗΜΑ 1.8: ELEMENT: SCROLLING

1.2. scrollLeft

Browser psounis

Το property scrollLeft:

- **Σε στοιχείο:**

Property	Επεξήγηση
scrollLeft	read: Πλήθος Pixels που έχει γίνει scroll οριζόντια set: Θέτοντας τιμή (σε px) γίνεται scroll σε αυτό το πλάτος

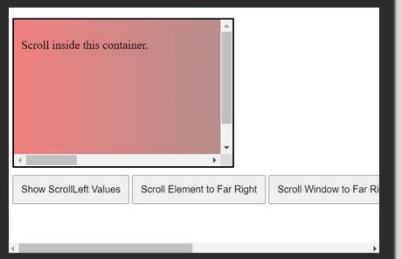
- **Σε παράθυρο**(Αντίστοιχη ιδιότητα στην οποία έχουμε πρόσβαση με δύο τρόπους):

Property	Επεξήγηση
scrollX (του window)	read: Πλήθος Pixels που έχει γίνει scroll οριζόντια
document.documentElement.scrollLeft	Η ιδιότητα αυτή είναι χρήσιμη για παρωχημένους browsers (προτιμούμε την ιδιότητα scrollX)

Παράδειγμα 2: scrollLeft.html

```html
<div class="scrollable-container" id="scrollableContainer">
  <div class="content">
    <p>Scroll inside this container.</p>
  </div>
</div>

<button id="showScrollLeftBtn">Show ScrollLeft Values</button>
<button id="scrollElementLeftBtn">Scroll Element to Far Right</button>
<button id="scrollWindowLeftBtn">Scroll Window to Far Right</button>
```

```javascript
// Get the button elements
const showScrollLeftBtn = document.getElementById('showScrollLeftBtn');
const scrollElementLeftBtn = document.getElementById('scrollElementLeftBtn');
const scrollWindowLeftBtn = document.getElementById('scrollWindowLeftBtn');
const container = document.getElementById('scrollableContainer');

// Event listener to show the scrollLeft values for both the window and the element
showScrollLeftBtn.addEventListener('click', () => {
  const windowScrollLeft = window.scrollX ||
document.documentElement.scrollLeft;
  const elementScrollLeft = container.scrollLeft;
  console.log(`Window scrollLeft: ${windowScrollLeft}`);
  console.log(`Element scrollLeft: ${elementScrollLeft}`);
});

// Event listener to scroll the container to the far right
scrollElementLeftBtn.addEventListener('click', () => {
  container.scrollLeft = container.scrollWidth;
  console.log('Scrolled element to far right');
});

// Event listener to scroll the window to the far right
scrollWindowLeftBtn.addEventListener('click', () => {
  window.scrollTo({ // doesn't scroll by setting window.scrollX
    left: document.body.scrollWidth,
    behavior: 'smooth'
  });
  console.log('Scrolled window to far right');
});
```



ΜΑΘΗΜΑ 1.8: ELEMENT: SCROLLING

2.1. Οι μέθοδοι scroll*()

Browser psounis

Μέθοδοι για Scrolling:

- Με τις παρακάτω μεθόδους μπορούμε να κάνουμε scrolling στο στοιχειο:

Μέθοδος	Επεξήγηση
scroll(x,y)	x: Scrolling κατά τον x-άξονα y: Scrolling κατά τον y-άξονα (σε σχέση με την αρχή)
scrollBy(x,y)	x: Scrolling κατά τον x-άξονα y: Scrolling κατά τον y-άξονα (σε σχέση με την τρέχουσα θέση)
scrollTo(x,y)	Ίδια με την scroll(x,y)

- Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε την εκδοχή των παραπάνω μεθόδων:

Μέθοδος	Επεξήγηση
scroll*(object)	object είναι αντικείμενο με μέλη: top: αντιστοιχεί στον x-άξονα, left: αντιστοιχεί στον y-άξονα, behavior: Παίρνει τιμές 'auto' και 'smooth' (που είναι αισθητικά πιο ευχάριστη)

Σημείωση:

- Οι παραπάνω μέθοδοι εμπίπτουν και στο window (θα το δούμε αναλυτικά στο μάθημα 3.1: window)

Παράδειγμα 3: scroll.html

```html
<h1>scroll() Example for Element</h1>
<p>Scroll the container both vertically and horizontally:</p>

<div class="scrollable-container" id="scrollableContainer">
  <div class="content">
    <p>Scroll inside this container.</p>
  </div>
</div>

<button id="scrollElementBtn">Scroll Element to (300, 300)</button
```

```javascript
const scrollElementBtn = document.getElementById('scrollElementBtn');
const container = document.getElementById('scrollableContainer');

// Scroll the element to the specified coordinates
scrollElementBtn.addEventListener('click', () => {
    container.scroll({
        top: 300,
        left: 300,
        behavior: 'smooth' // Smooth scrolling effect
    });
    console.log('Scrolled element to (300, 300)');
});
```

ΜΑΘΗΜΑ 1.8: ELEMENT: SCROLLING

2.2. scrollIntoView()

Browser psounis

Η μέθοδος scrollIntoView():

- Η ακόλουθη ισχυρή μέθοδος φέρνει το στοιχείο στο viewport του browser:

Μέθοδος	Επεξήγηση
scrollIntoView([options])	Φέρνει το στοιχείο στην ορατή περιοχή του browser (έστω και μερικώς).

- Το options είναι αντικείμενο με τα εξής προαιρετικά μέλη:
  - behavior (default: 'auto', η τιμή 'smooth' κάνει την μετάβαση πιο ευχάριστη)
  - block: Κάθετη στοίχιση στο viewport. Τιμές: 'start' (default), 'center', 'end'
  - inline: Οριζόντια στοίχιση. Τιμές: 'start', 'center', 'end'

Παράδειγμα 4: scrollIntoView.html

```html
<div class="navbar">
  <a data-target="section1">Section 1</a>
  <a data-target="section2">Section 2</a>
  <a data-target="section3">Section 3</a>
  <a data-target="section4">Section 4</a>
</div>

<div id="section1" class="section">Welcome to Section 1</div>
<div id="section2" class="section">Welcome to Section 2</div>
<div id="section3" class="section">Welcome to Section 3</div>
<div id="section4" class="section">Welcome to Section 4</div>
```

```javascript
// Select all navigation links
document.querySelectorAll('.navbar a').forEach(link => {
  link.addEventListener('click', () => {
    // Get the target section based on the data-target attribute
    const targetId = link.getAttribute('data-target');
    const targetElement = document.getElementById(targetId);

    // Scroll the target section into view with a smooth behavior
    targetElement.scrollIntoView({ behavior: 'smooth', block: 'start' });

    // Optional: Highlight the target section briefly
    targetElement.classList.add('highlight');
    setTimeout(() => targetElement.classList.remove('highlight'), 1000);
  });
});
```

ΜΑΘΗΜΑ 1.8: ELEMENT: SCROLLING

3. Το event scroll

Το event scroll:

- Το ακόλουθο event ενεργοποιείται συνεχώς όσο ο χρήστης κάνει scroll

event	Επεξήγηση
scroll	Πυροδοτείται όταν ο χρήστης κάνει scroll με το ποντίκι στο στοιχείο

Παρατήρηση:

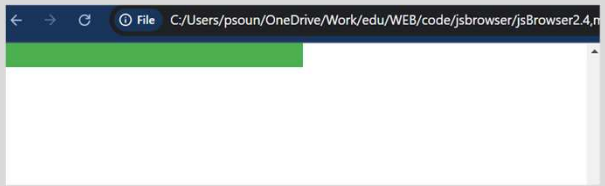
- Το παραπάνω event μπορεί να ενεργοποιηθεί και στο αντικείμενο window (βλ. μάθημα 2.4)

Παράδειγμα 5: scroll-event.html

```html
<h1>Scroll Event for Element</h1>
<p>Scroll inside the container:</p>

<div class="scrollable-container" id="scrollableContainer">
 <div class="content">
  <p>Scroll inside this container to see the effect.</p>
 </div>
</div>
```

```css
.scrollable-container {
  width: 300px;
  height: 200px;
  overflow: auto;
  border: 2px solid black;
  margin-top: 20px;
}
.content {
  height: 800px;
  background: linear-gradient(to bottom, lightblue, lightgreen);
  padding: 10px;
}
```

```js
const container = document.getElementById('scrollableContainer');

// Event listener for the scroll event on the container
container.addEventListener('scroll', () => {
  const scrollTop = container.scrollTop;
  console.log(`Element scrollTop: ${scrollTop}`);
});
```



4.1. Παράδειγμα: Εφέ σε Sticky Header

Εφέ σε sticky header

- Ένα κλασικό εφέ σε header:
  - Αρχικά εμφανίζεται ένα header το οποίο αλλάζει όταν ο χρήστης κάνει scroll στην σελίδα
  - Όταν επανέλθει στην αρχική θέση, τότε εμφανίζεται πάλι το αρχικό header

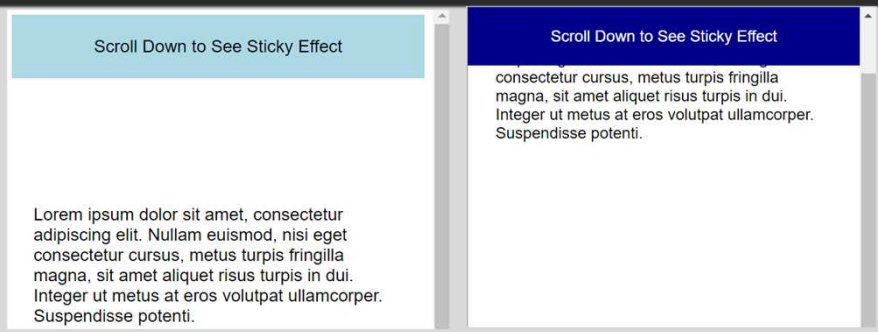
Παράδειγμα 6: example-sticky-header.html

```html
<div class="header" id="header">Scroll Down to See Sticky Effect</div>
<div class="content">
 <p>Lorem...</p>
 ...
/div>
```

```css
body {
   font-family: Arial, sans-serif;
}
.header {
   padding: 20px;
   background-color: lightblue;
   text-align: center;
   position: relative;
   transition: background-color 0.3s
ease;
}
```

```css
.header.sticky {
   position: fixed;
   top: 0;
   left: 0;
   right: 0;
   background-color: darkblue;
   color: white;
}
.content {
   padding: 100px 20px;
}
```

```javascript
const header = document.getElementById('header');
const headerOffsetTop = header.offsetTop; // The initial top position of the header

window.addEventListener('scroll', function() {
  const scrollPosition = window.scrollY;

  if (scrollPosition > headerOffsetTop) {
    header.classList.add('sticky');
  } else {
    header.classList.remove('sticky');
  }
});
```



Σημείωση:

- Χρησιμοποιούμε το scroll event του αντικειμένου window
- Πυροδοτείται όταν ο χρήστης κάνει scroll στη σελίδα (περισσότερα στην ενότητα 2)

4.2. Παράδειγμα: Lazy Loading

Lazy Loading:

- Φορτώνονται καινούργια αντικείμενα όσο ο χρήστης σκρολλάρει (**doomscrolling**)

Παράδειγμα 7: example-lazy-loading.html

```html
<div class="scrollable-container" id="scrollableContainer">
  <div id="content">
    <!-- Initial items -->
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
    <div class="item">Item 4</div>
    <div class="item">Item 5</div>
  </div>
  <div id="loadingIndicator">Loading more items...</div>
</div>
```

```css
.scrollable-container {
    width: 400px;
    height: 300px;
    overflow: auto;
    border: 2px solid black;
    margin-top: 20px;
}
.item {
    padding: 15px;
    margin: 10px;
    background-color: lightgray;
    border-radius: 5px;
    height: 300px;
}
```

```css
#loadingIndicator {
    text-align: center;
    padding: 10px;
    display: none;
    color: gray;
}
```

```javascript
const container = document.getElementById('scrollableContainer');
const content = document.getElementById('content');
const loadingIndicator = document.getElementById('loadingIndicator');
let itemCount = 5; // Track number of items loaded
let isLoading = false;
// Function to simulate loading more content
function loadMoreContent() {
    if (isLoading) return; // Prevent multiple loading events
    isLoading = true;
    loadingIndicator.style.display = 'block';
    setTimeout(() => {
        // Simulate content loading
        for (let i = 0; i < 5; i++) {
            const newItem = document.createElement('div');
            newItem.className = 'item';
            newItem.textContent = `Item ${++itemCount}`;
            content.appendChild(newItem);
        }
        loadingIndicator.style.display = 'none';
        isLoading = false;
    }, 1000); // Simulate a 1-second loading time
}
// Event listener for scrolling
container.addEventListener('scroll', () => {
    // Calculate the bottom position
    const bottomReached = container.scrollTop + container.clientHeight >=
                          container.scrollHeight - 10;
    if (bottomReached) {
        loadMoreContent();
    }
});
```

4.3. Παράδειγμα: Chat Window

Chat Window:

- Σε ένα παράθυρο chat, πρέπει αυτόματα να γίνεται scroll στο πιο πρόσφατο μήνυμα

Παράδειγμα 8: example-chat.html

```html
<h1>Chatbox Scrolling Example</h1>
<div class="chatbox" id="chatbox">
  <div class="message">Welcome to the chat!</div>
  <div class="message">This is the first message.</div>
  <div class="message">Feel free to add more messages.</div>
</div>

<div class="input-area">
  <input type="text" id="messageInput" placeholder="Type a message...">
  <button id="sendButton">Send</button>
</div>
```

```css
.chatbox {
    width: 400px;
    height: 200px;
    border: 2px solid black;
    overflow-y: auto;
    padding: 10px;
    margin-top: 20px;
}
.message {
    padding: 8px;
    margin: 5px 0;
    background-color: lightgray;
    border-radius: 5px;
}
```

```css
.input-area {
    margin-top: 10px;
}

.input-area input {
    width: 300px;
    padding: 8px;
}

.input-area button {
    padding: 8px;
    cursor: pointer;
}
```

```javascript
const chatbox = document.getElementById('chatbox');
const messageInput = document.getElementById('messageInput');
const sendButton = document.getElementById('sendButton');

// Function to add a new message to the chatbox
function addMessage(text) {
    const newMessage = document.createElement('div');
    newMessage.className = 'message';
    newMessage.textContent = text;
    chatbox.appendChild(newMessage);

    // Automatically scroll to the bottom
    chatbox.scrollTop = chatbox.scrollHeight;
}

// Event listener for the Send button
sendButton.addEventListener('click', () => {
    const messageText = messageInput.value.trim();
    if (messageText !== '') {
        addMessage(messageText);
        messageInput.value = ''; // Clear the input
    }
});

// Optionally, allow pressing "Enter" to send the message
messageInput.addEventListener('keydown', (event) => {
    if (event.key === 'Enter') {
        sendButton.click();
    }
});
```

ΜΑΘΗΜΑ 1.8: ELEMENT: SCROLLING

4.4. Παράδειγμα: Active Section

Active Section:

- Εντοπίζεται το μέρος που είναι ενεργό, ώστε να ενημερώνεται το μενού πλοήγησης

Παράδειγμα 9: active-section.html

```html
<div class="navbar">
  <a href="#section1" id="link1">Section 1</a>
  <a href="#section2" id="link2">Section 2</a>
  <a href="#section3" id="link3">Section 3</a>
  <a href="#section4" id="link4">Section 4</a>
</div>
<section id="section1">
  <h2>Section 1</h2>
  <p>Content for Section 1...</p>
</section>
...
```

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}
.navbar {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    background: #333;
    color: white;
    display: flex;
    justify-content: space-around;
    padding: 10px 0;
    z-index: 1000;
}
```

```css
.navbar a {
    color: white;
    text-decoration: none;
    padding: 8px 16px;
}
.navbar a.active {
    background-color: #555;
    border-radius: 4px;
}
section {
    height: 100vh;
    padding-top: 50px;
    margin-top: 50px;
    border: 1px solid #ccc;
    background: linear-gradient(to bottom right, #f0f0f0, #d3d3d3);
}
```

```javascript
const sections = document.querySelectorAll('section');
const navLinks = document.querySelectorAll('.navbar a');

// Function to highlight the active navigation link based on the current section
function highlightActiveSection() {
  let currentSectionIndex = -1;

  // Loop through sections to find the one currently in view
  sections.forEach((section, index) => {
    const rect = section.getBoundingClientRect();
    if (rect.top <= 100 && rect.bottom >= 100) {
      currentSectionIndex = index;
    }
  });

  // Remove the active class from all links
  navLinks.forEach(link => link.classList.remove('active'));

  // Add the active class to the current section's link
  if (currentSectionIndex !== -1) {
    navLinks[currentSectionIndex].classList.add('active');
  }
}

// Event listener for the scroll event on the window
window.addEventListener('scroll', highlightActiveSection);

// Initial call to set the active link based on the current scroll position
highlightActiveSection();
```