ΠΕΡΙΕΧΟΜΕΝΑ:

1. Timeouts
    1. Η μέθοδος setTimeout()
    2. clearTimeout()
2. Intervals
3. Παραδείγματα
    1. JS Animation
    2. REST API Calls

ΜΑΘΗΜΑ 3.3: Μέθοδοι Χρονισμού

1. Η μέθοδος setTimeout()

Συναρτήσεις Χρονισμού:

- Προγραμματίζουν την εκτέλεση κώδικα μετά από κάποιο χρονικό διάστημα
- Πολύ χρήσιμες για την εκτέλεση ασύγχρονων ενεργειών και ενεργειών που πρέπει να εκτελούνται περιοδικά, όπως animations, notifications και ανάσυρση δεδομένων από το server.

Η μέθοδος setTimeout():

- Η ακόλουθη μέθοδος ανήκει στο αντικείμενο **window**:

Μέθοδος	Επεξήγηση
setTimeout (function, delay, …args)	Εκτελεί τη συνάρτηση function, μετά από το χρονικό διάστημα delay (σε ms) …args: Ορίσματα σε συνάρτηση

Σημείωση:

- Η μέθοδος είναι ασύγχρονη, non-blocking, δηλαδή:
  - Όταν καλείται, το κύριο νήμα συνεχίζει την εκτέλεση του προγράμματος, όσο ο timer συνεχίζει την μέτρηση του χρόνου
  - Όταν ο χρόνος του delay παρέλθει, η συνάρτηση μπαίνει στην callback queue του event loop και θα εκτελεστεί μόνο όταν η call stack είναι άδεια, όπως είδαμε στα μαθήματα 2.*

Παράδειγμα 1: setTimeout

```
// Named function to display a message with a countdown
function showCountdown(seconds) {
  console.log(`Countdown starting for ${seconds} seconds...`);

  setTimeout(function() {
    console.log(`${seconds} seconds passed!`);
  }, seconds*1000);
}

// Arrow function to display a reminder after a delay
function showReminder(message, seconds) {
  console.log(`Reminder set after ${seconds} seconds...`);

  setTimeout((message) => {
    console.log(`Reminder: ${message}`);
  }, seconds*1000, message);
}

// Start countdown
showCountdown(10);
// Show reminder
showReminder("Time passes!", 5);
```

```
Countdown starting for 10 seconds...
Reminder set after 5 seconds...
Reminder: Time passes!
10 seconds passed!
```

clearTimeout:

- Με την μέθοδο του **window**:

Μέθοδος	Επεξήγηση
clearTimeout(id)	Ακυρώνει ένα timeout το οποίο έχει καθοριστεί από την setTimeout()

  - το id που τίθεται ως όρισμα έχει επιστραφεί από την setTimeout()
    - οπότε πρέπει να το αποθηκεύσουμε για να το χρησιμοποιήσουμε στην clearTimeout()

Παράδειγμα 2: clearTimeout.html

```
let id = setTimeout(function() {
   console.log("will never happen");
}, 1000);

clearTimeout(id);
```

Παράδειγμα 2: clicking-timer

```
let timeoutID;
let timeLimit = 5000; // 5 seconds in milliseconds

const button = document.getElementById('click-button');
const messageDisplay = document.getElementById('message');

// Function to start or reset the timer
function startTimer() {
   clearTimeout(timeoutID); // Clear any previous timeout

   // Set a new timeout for 5 seconds
   timeoutID = setTimeout(() => {
      messageDisplay.textContent = "Too slow! You didn't click in time.";
      button.disabled = true; // Disable the button when time runs out
   }, timeLimit); // 5 seconds
}

// Event listener to reset the timer when the button is clicked
button.addEventListener('click', () => {
   messageDisplay.textContent = 'Good job! You clicked in time.';
   startTimer(); // Reset the timer
});

// Start the initial timer when the page loads
startTimer();
```

ΜΑΘΗΜΑ 3.3: Μέθοδοι Χρονισμού

2. Intervals

Intervals:

- Τρέχει κώδικα ανά χρονικό διάστημα (και όχι μόνο μία φορά όπως η setTimeout())

Μέθοδος	Επεξήγηση
setInterval (function, delay, …args)	Τρέχει την συνάρτηση function, ανά χρονικό διάστημα (delay) …args: Ορίσματα της συνάρτησης function Επιστρέφει id του interval
clearInterval(id)	Ακυρώνει την εκτέλεση του interval

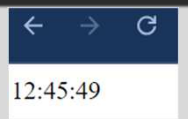
Σημείωση:

- Οι παραπάνω μέθοδοι είναι αρκετά σημαντικές και με αυτές μπορούμε να κατασκευάσουμε λειτουργίες που βλέπουμε συχνά σε σελίδες:
  - Ρολόγια πραγματικού χρόνου
  - Επαναλήψεις animations
  - Ενημέρωση κάποιων δεδομένων από server

Παράδειγμα 3: clock

```html
<div id="clock"></div>
```

```js
function updateClock() {
  const clockElement = document.getElementById('clock');

  // Get the current time
  const now = new Date();

  // Extract hours, minutes, and seconds
  const hours = String(now.getHours()).padStart(2, '0'); // pad with 0 if necessary
  const minutes = String(now.getMinutes()).padStart(2, '0');
  const seconds = String(now.getSeconds()).padStart(2, '0');

  // Format the time string
  const currentTime = `${hours}:${minutes}:${seconds}`;

  // Display the time on the page
  clockElement.textContent = currentTime;
}

// Update the clock every second (1000 milliseconds)
setInterval(updateClock, 1000);

// Initial call to set the time immediately without waiting 1 second
updateClock();
```



Παράδειγμα 4: animation

```html
<div id="box"></div>
```

```css
#box {
  width: 50px;
  height: 50px;
  background-color: red;
  position: absolute;
  top: 100px;
  left: 0;
}
```

```javascript
let position = 0;
const box = document.getElementById('box');

// Function to move the box
function moveBox() {
  // Move the box 5px to the right
  position += 5;
  box.style.left = position + 'px';

  // Reset the position once it reaches 500px
  if (position >= 500) {
    position = 0;
  }
}

// Set an interval to repeat the animation every 50ms (creates a smooth animation)
setInterval(moveBox, 50);
```

Σημείωση:

- Αντίστοιχο αποτέλεσμα μπορούμε να πετύχουμε και με τη CSS (που ίσως είναι καλύτερο, αφού δεν επιβαρύνει το νήμα της JS με έξτρα κλήσεις μεθόδων)

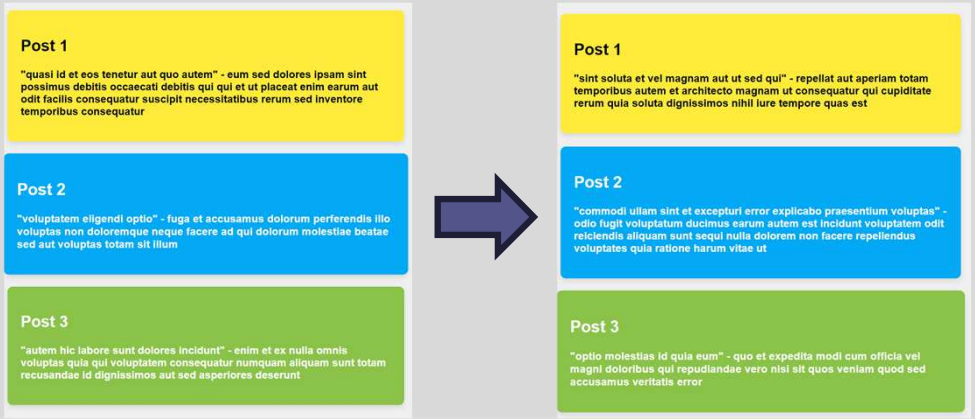
Παράδειγμα 5: animationCSS

```html
<div id="box"></div>
```

```css
#box {
  width: 50px;
  height: 50px;
  background-color: red;
  position: absolute;
  top: 100px;
  left: 0;
  animation: move 5s linear infinite;
}

@keyframes move {
  0% {
    left: 0;
  }
  100% {
    left: 500px;
  }
}
```

Παράδειγμα 6: rest-api

```html
<div id="posts">
 <div class="post-box" id="post1">
  <h2>Post 1</h2>
  <p class="post-description">Fetching post 1...</p>
 </div>

 <div class="post-box" id="post2">
  <h2>Post 2</h2>
  <p class="post-description">Fetching post 2...</p>
 </div>

 <div class="post-box" id="post3">
  <h2>Post 3</h2>
  <p class="post-description">Fetching post 3...</p>
 </div>
</div>
```




```javascript
const postElements = [
  document.querySelector('#post1 .post-description'),
  document.querySelector('#post2 .post-description'),
  document.querySelector('#post3 .post-description')
];

// Function to fetch a random post and update the respective box
function fetchPost(index) {
  fetch('https://jsonplaceholder.typicode.com/posts')
    .then(response => response.json())
    .then(data => {
      // Pick a random post from the fetched data
      const randomPost = data[Math.floor(Math.random() * data.length)];
      const title = randomPost.title;
      const body = randomPost.body;

      // Update the corresponding post box with the new post
      postElements[index].textContent = `"${title}" - ${body}`;
    })
    .catch(error => {
      console.error('Error fetching post:', error);
    });
}

// Function to fetch all posts
function fetchAllPosts() {
  for (let i = 0; i < postElements.length; i++) {
    fetchPost(i);
  }
}

// Fetch posts immediately when the page loads
fetchAllPosts();

// Fetch new posts every 3 seconds
setInterval(fetchAllPosts, 3000);
```