

# JS Browser

## ΜΑΘΗΜΑ 3.6

## STORAGE

### ΠΕΡΙΕΧΟΜΕΝΑ:

1. Storage API
  1. Διαγραφή-Ενημέρωση Ζευγαριών
  2. Επανάληψη και Event
  3. Χειρισμός Λαθών και Αποθήκευση ως JSON
2. Παραδείγματα
  1. User Preferences
  2. Cart
  3. Φόρμα Πολλών Βημάτων

### Storage API:

- Υλοποιείται από τα **window.localStorage** και **window.sessionStorage**
- Επιτρέπει να αποθηκεύουμε δεδομένα στον browser που αφορούν τη σελίδα μας:
  - Έτσι δεν απαιτείται να ανταλλάσσουμε πολλά δεδομένα με τον server, αφού σε αυτά μπορούμε να αποθηκεύσουμε ακόμη και ~5-10MB
  - Περιορίζοντας τη χρήση των cookies και κάνοντας την εφαρμογή μας πιο αποδοτική.

### Local Storage:

- Τα δεδομένα **παραμένουν ακόμη κι όταν κλείσει ο browser.**
- Οπότε σε αυτήν μπορούμε να αποθηκεύσουμε δεδομένα όπως:
  - Προτιμήσεις χρήστη (π.χ. γλώσσα και theme)
  - Authentication Tokens
  - Προϊόντα σε ένα καλάθι αγορών κ.λπ.

### Session Storage:

- Τα δεδομένα **σβήνονται όταν κλείσει ο browser.**
- Οπότε σε αυτήν μπορούμε να αποθηκεύσουμε δεδομένα όπως:
  - Δεδομένα φόρμών που συμπληρώνονται σε πολλά βήματα.
  - Προσωρινά tokens.
  - κ.λπ.

### Αποθήκευση/Ανάσυρση Δεδομένων:

- Τα δεδομένα αποθηκεύονται ως key/value pairs με τις μεθόδους:

Μέθοδος	Επεξήγηση
setItem(key, value)	Αποθηκεύει το ζευγάρι key: value
getItem(key)	Επιστρέφει το value του key

### Παράδειγμα 1: storage

```
console.log(localStorage.getItem("local"));
console.log(sessionStorage.getItem("session"));

localStorage.setItem("local", "1");
sessionStorage.setItem("session", "2");
```

1<sup>η</sup> εκτέλεση:

```
null
null
```

Refresh:

```
1
2
```

Κλείσιμο και Άνοιγμα Tab:

```
1
null
```

### Διαγραφή ζευγαριών:

- Για να διαγράψουμε δεδομένα από την localStorage ή την sessionStorage χρησιμοποιούμε τις μεθόδους:

Μέθοδος	Επεξήγηση
removeItem(key)	Διαγραφή ζεύγους με κλειδί key
clear()	Διαγραφή όλων των ζευγών

### Ενημέρωση ζεύγους:

- Χρησιμοποιούμε την setItem() αν θέλουμε να ενημερώσουμε την τιμή κάποιου κλειδιού.

### Παράδειγμα 2: remove

```
for (let i=1; i<=10; i++)
    localStorage.setItem("local"+ i, i);

console.log(localStorage.length);

localStorage.removeItem("local6");
localStorage.setItem("local1", 11);

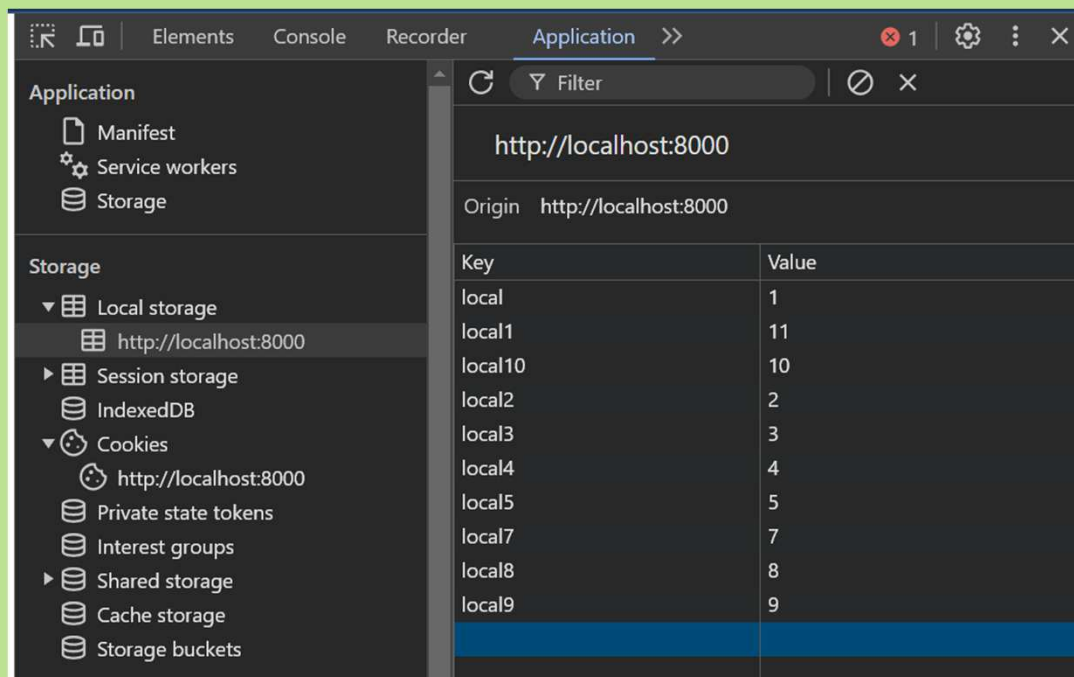
console.log(localStorage.length);
```

11

10

### Storage στον Google Chrome:

- Στα Developer Tools βρίσκουμε στην καρτέλλα Application τα Local Storage και Session Storage:



- Όπου μπορούμε να διαγράψουμε εγγραφές, να προσθέσουμε νέες (με δεξί κλικ) κ.α.

### Επανάληψη:

- Η ακόλουθη μέθοδος μπορεί να χρησιμοποιηθεί για να κάνουμε επανάληψη επί των κλειδιών που έχουν αποθηκευτεί:

Property	Επεξήγηση
key(index)	Επιστρέφει το κλειδί που έχει αποθηκευτεί με την εσωτερική αρίθμηση index

### Παράδειγμα 3: loop

```
for (let i=0; i<localStorage.length; i++) {
  let key = localStorage.key(i)
  console.log(key, localStorage.getItem(key))
}
```



```
local11 11
local13 3
local14 4
local18 8
local12 2
local15 5
local19 9
local10 10
local17 7
local 1
```

### Event storage:

- Πυροδοτείται όταν πραγματοποιηθεί ενέργεια σε κάποια από τις storage (είτε localStorage είτε sessionStorage) από κάποια διαφορετική σελίδα στον ίδιο server
- Το αντικείμενο event περιέχει πολλές χρήσιμες πληροφορίες:

Property	Επεξήγηση
key	Το κλειδί που άλλαξε
oldValue	Παλιά τιμή
newValue	Νέα τιμή
url	Το url που άλλαξε την storage
storageArea	Reference σε localStorage ή sessionStorage

### Παράδειγμα 4: event και eventListener

```
localStorage.setItem("local1", 111);
localStorage.removeItem("local1");
```




```
window.addEventListener('storage', (event) => {
  console.log('Key changed:', event.key);
  console.log('Old value:', event.oldValue);
  console.log('New value:', event.newValue);
  console.log('URL of change:', event.url);
  console.log('Storage area:', event.storageArea);
});
```



Χειρισμός Λαθών:


- Αν υπερβούμε τον διαθέσιμο χώρο, τότε προκαλείται εξαίρεση QuotaExceededError (οπότε σε εκτενή χρήση των Storage, θα πρέπει να το προβλέπουμε)

Παράδειγμα 5: error

```
// Function to trigger a QuotaExceededError by saving large data
function triggerQuotaError() {
  try {
    // Create a large string to exceed storage limits (6MB)
    const largeData = 'x'.repeat(1024 * 1024 * 6); // 6MB string
    localStorage.setItem('largeData', largeData); // Try to save in
    localStorage
    document.getElementById('message').innerText = 'Large data saved
    successfully!';
  } catch (e) {
    if (e.name === 'QuotaExceededError') {
      document.getElementById('message').innerText = 'Storage limit
      exceeded!';
      console.error('QuotaExceededError:', e);
    } else {
      document.getElementById('message').innerText = 'Error accessing
      storage.';
      console.error('Storage error:', e);
    }
  }
}
```

Αποθήκευση με JSON:

- Συχνά χρησιμοποιείται JSON για να αποθηκεύουμε αντικείμενα στις Storage
- Έχουμε το αντικείμενο που θέλουμε να αποθηκεύσουμε και:
  - Καλούμε την JSON.stringify() για να αποθηκευτεί ως συμβολοσειρά
  - Καλούμε την JSON.parse(object), για να επαναφέρουμε το αντικείμενο από την storage

Παράδειγμα 6: json

```
// Function to save data as a JSON object in localStorage
function saveUserData() {
  const username = document.getElementById('username').value;
  const age = document.getElementById('age').value;
  const user = {
    name: username,
    age: age
  };
  // Save the object as a JSON string in localStorage
  localStorage.setItem('userData', JSON.stringify(user));
  document.getElementById('message').innerText = 'User data saved!';
}

// Function to load data from localStorage
function loadUserData() {
  // Get the stored JSON string
  const userData = localStorage.getItem('userData');
  // Parse the JSON string back into an object
  if (userData) {
    const user = JSON.parse(userData);
    document.getElementById('message').innerText = `Username: ${user.name}, Age:
    ${user.age}`;
  } else {
    document.getElementById('message').innerText = 'No user data found.';
  }
}
```

### Παράδειγμα: User Preferences

- Οι προτιμήσεις χρήστη (π.χ. theme και γλώσσα) δεν αλλάζουν και επιλέγουμε να αποθηκευτούν στην localStorage

### Παράδειγμα 7: example-preferences

```
<div class="container">
  <h1>User Preferences (Theme & Language)</h1>
```

```
  <!-- Theme Preference -->
  <h2>Choose Theme</h2>
  <button onclick="setTheme('light')">Light Mode</button>
  <button onclick="setTheme('dark')">Dark Mode</button>
```

```
  <!-- Language Preference -->
  <h2>Choose Language</h2>
  <button onclick="setLanguage('English')">English</button>
  <button onclick="setLanguage('Greek')">Greek</button>
  <p id="languageDisplay"></p>
</div>
```

```
body {
  font-family: Arial, sans-serif;
  transition: background-color 0.3s;
}

.light-mode {
  background-color: #fff;
  color: #000;
}

.dark-mode {
  background-color: #333;
  color: #fff;
}

.container {
  ...
}
```

```
// Function to set and save the theme in localStorage
function setTheme(theme) {
  localStorage.setItem('theme', theme);
  applyTheme();
}

// Function to apply theme based on localStorage value
function applyTheme() {
  const theme = localStorage.getItem('theme');
  if (theme) {
    document.body.className = theme === 'dark' ? 'dark-mode' : 'light-mode';
  } else {
    console.error("No theme set or corrupted data.");
  }
}

// Function to set and save the language preference
function setLanguage(language) {
  try {
    localStorage.setItem('language', language);
    displayLanguage();
  } catch (error) {
    console.error("Error saving language preference:", error);
  }
}

// Function to display the saved language preference
function displayLanguage() {
  const language = localStorage.getItem('language');
  if (language) {
    document.getElementById('languageDisplay').innerText = "Language: " + language;
  } else {
    console.error("No language set or corrupted data.");
  }
}

// Apply theme and display stored language on page load
window.onload = function() {
  applyTheme();
  displayLanguage();
}
```

### Παράδειγμα: Cart

- Επιλέγουμε για την εμπειρία χρήστη, ένα καλάθι αγορών που δεν έχει αγοραστεί, να αποθηκεύεται στην localStorage

### Παράδειγμα 8: example-cart



```
<div class="container">
  <h1>Shopping Cart</h1>

  <!-- Shopping Cart -->
  <h2>Add Items to Cart</h2>
  <input type="text" id="cartItem" placeholder="Add item to cart">
  <button onclick="addToCart()">Add to Cart</button>

  <div class="cart">
    <h3>Cart Items:</h3>
    <ul id="cartList"></ul>
  </div>
</div>
```



```
body {
  font-family: Arial, sans-serif;
}

.container {
  max-width: 600px;
  margin: 50px auto;
  padding: 20px;
  text-align: center;
}

.cart {
  margin-top: 20px;
}
```



// Function to add items to the shopping cart and store in localStorage

```
function addToCart() {
  const item = document.getElementById('cartItem').value;
  if (!item) return; // Do nothing if input is empty

  let cart = [];
  try {
    // Get existing cart items from localStorage or start a new cart
    cart = JSON.parse(localStorage.getItem('cart')) || [];
    cart.push(item);
    // Save the updated cart back to localStorage
    localStorage.setItem('cart', JSON.stringify(cart));
    // Clear the input field
    document.getElementById('cartItem').value = '';
    // Display updated cart
    displayCart();
  } catch (error) {
    console.error("Error adding to cart:", error);
  }
}
```



// Function to display cart items from localStorage

```
function displayCart() {
  const cart = JSON.parse(localStorage.getItem('cart')) || [];
  const cartList = document.getElementById('cartList');
  cartList.innerHTML = ''; // Clear current list

  cart.forEach((item, index) => {
    const li = document.createElement('li');
    li.textContent = `${index + 1}. ${item}`;
    cartList.appendChild(li);
  });
}

// Display stored cart items on page load
window.onload = function() {
  displayCart();
}
```



**Παράδειγμα: Φόρμα Πολλών Βημάτων**

- Εδώ επιλέγουμε sessionStorage, ώστε τα δεδομένα του χρήστη σε κάποιο βήμα, να παραμένουν αν κάνει Back/Forward

**Παράδειγμα 9: example-forms**

```
<div class="container">
  <h1>Two-Step Form</h1>

  <!-- Step 1 -->
  <div class="step" id="step1">
    <h2>Step 1: Personal Information</h2>
    <label for="name">Name:</label>
    <input type="text" id="name" placeholder="Enter your name">
    <br>
    <label for="email">Email:</label>
    <input type="email" id="email" placeholder="Enter your email">
  </div>

  <!-- Step 2 -->
  <div class="step" id="step2">
    <h2>Step 2: Review & Submit</h2>
    <p><strong>Name:</strong> <span id="reviewName"></span></p>
    <p><strong>Email:</strong> <span id="reviewEmail"></span></p>
  </div>

  <!-- Navigation Buttons -->
  <div class="buttons">
    <button onclick="prevStep()" id="prevBtn">Previous</button>
    <button onclick="nextStep()" id="nextBtn">Next</button>
  </div>
</div>
```



```
let currentStep = 1;
// Function to show the current step and hide the others
function showStep(step) {
  document.querySelectorAll('.step').forEach((el) => el.classList.remove('active'));
  document.getElementById('step' + step).classList.add('active');
  // Populate review fields in step 2
  if (step === 2) {
    reviewData(); // Populate the review section when user reaches step 2
  }
}
// Function to move to the next step
function nextStep() {
  saveStepData();
  ...
}
// Function to move to the previous step
function prevStep() {
  saveStepData();
  ...
}
// Function to save current step data to sessionStorage
function saveStepData() {
  if (currentStep === 1) {
    sessionStorage.setItem('name', document.getElementById('name').value);
    sessionStorage.setItem('email', document.getElementById('email').value);
  }
}
// Function to review data in Step 2
function reviewData() {
  document.getElementById('reviewName').innerText = sessionStorage.getItem('name') || '';
  document.getElementById('reviewEmail').innerText = sessionStorage.getItem('email') || '';
}
// On page load, restore the step data from sessionStorage and show the first step
window.onload = function() {
  document.getElementById('name').value = sessionStorage.getItem('name') || '';
  document.getElementById('email').value = sessionStorage.getItem('email') || '';
  showStep(currentStep);
}
```

