ΠΕΡΙΕΧΟΜΕΝΑ:

ΜΑΘΗΜΑ 2.6: Event Delegation

1. Event Delegation

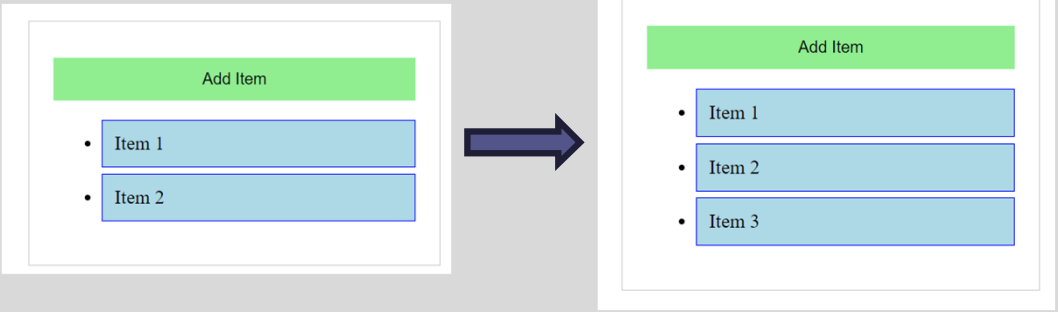
Σημείωση:

- Στο μάθημα 2.2 είδαμε τα properties του αντικειμένου event:

Property	Επεξήγηση
target	Το στοιχείο που προκάλεσε το event
currentTarget	Το στοιχείο του οποίου ο handler ενεργοποιήθηκε

- Μέσω αυτών μπορούμε να αναθέσουμε (**delegate**) την διαχείριση events σε γονικό στοιχείο, το οποίο θα διαχειρίζεται τα events που προκαλούνται σε παιδιά του.
- Είναι πολύ χρήσιμο γιατί μειώνει τον κώδικα των events (ιδίως όταν έχουμε προσθαφαίρεση στοιχείων)

Παράδειγμα 1: event-delegation.html



Παράδειγμα 1: events-drag-and-drop.html

```html
<div class="list-container">
  <button class="add-button">Add Item</button>
  <ul id="itemList">
    <li class="list-item">Item 1</li>
    <li class="list-item">Item 2</li>
  </ul>
</div>
```

```javascript
const listContainer = document.querySelector('.list-container');
const itemList = document.getElementById('itemList');
const addButton = document.querySelector('.add-button');

// Event listener attached to the container to handle clicks on list items
listContainer.addEventListener('click', (event) => {
    if (event.target.classList.contains('list-item')) {
        console.log(`You clicked on ${event.target.textContent}`);
        event.stopPropagation();
    }
});
addButton.addEventListener('click', () => {
    const newItem = document.createElement('li');
    newItem.className = 'list-item';
    newItem.textContent = `Item ${itemList.children.length + 1}`;
    itemList.appendChild(newItem);
});
// Example of another listener on the container to demonstrate propagation control
listContainer.addEventListener('click', () => {
    console.log('Container clicked'); // This will not log if stopPropagation() is called on the list item
});
```

ΜΑΘΗΜΑ 2.6: Event Delegation

1.1. Form Validation

Browser psounis

Form Validation:

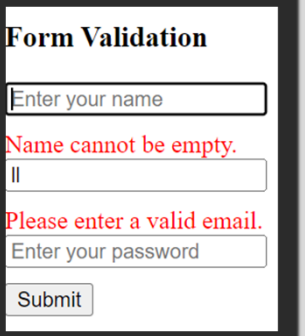
- Ένας μόνο listener (στο επίπεδο της φόρμας) επικυρώνει τα δεδομένα του χρήστη

Παράδειγμα 2: form-validation.html

```html
<form id="user-form">
  <input type="text" class="input-field" placeholder="Enter your name" data-type="name">
  <span class="error" id="name-error">Invalid input!</span>

  <input type="email" class="input-field" placeholder="Enter your email" data-type="email">
  <span class="error" id="email-error">Invalid input!</span>

  <input type="password" class="input-field" placeholder="Enter your password" data-type="password">
  <span class="error" id="password-error">Invalid input!</span>

  <button type="submit">Submit</button>
</form>
```

```javascript
const form = document.getElementById('user-form');
let formIsValid = true;

// Show or hide error message based on validity
function toggleError(errorId, isValid, errorMessage) {
    const errorElement = document.getElementById(errorId);
    if (isValid) {
        errorElement.style.display = 'none';
    } else {
        errorElement.textContent = errorMessage;
        errorElement.style.display = 'block';
    }
}
```

```javascript
// Event Listener for input validation
form.addEventListener('input', function(event) {
    const input = event.target;
    const inputType = input.getAttribute('data-type');
    let isValid = true;

    // Switch case for different input types
    switch (inputType) {
        case 'name':
            isValid = input.value.trim() !== '';
            toggleError('name-error', isValid, 'Name cannot be empty.');
            break;
        case 'email':
            isValid = input.value.includes('@');
            toggleError('email-error', isValid, 'Please enter a valid email.');
            break;
        case 'password':
            isValid = input.value.length >= 6;
            toggleError('password-error', isValid, 'Password must be at least 6 characters long.');
            break;
    }

    formIsValid = document.querySelectorAll('.error[style*="display: block"]').length === 0;
});

// Form submit event listener
form.addEventListener('submit', function(event) {
    if (!formIsValid) {
        event.preventDefault();
        console.log('Form submission prevented due to validation errors.');
    } else {
        console.log('Form submitted successfully!');
    }
});
```

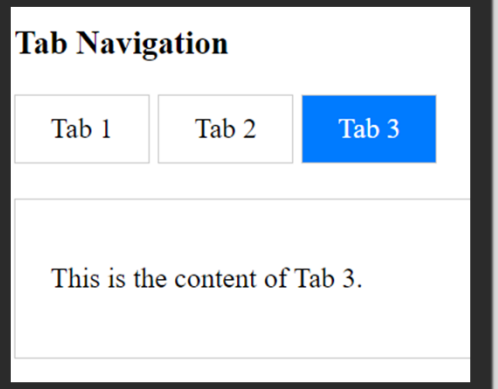


Menus & Tabs:

- Ένας μόνο listener (στο επίπεδο του container) ελέγχει τις αλληλεπιδράσεις με τον χρήστη

Παράδειγμα 3: tabs.html

```html
<h3>Tab Navigation</h3>
<div class="tabs" id="tabs">
 <div class="tab" data-content="content1">Tab 1</div>
 <div class="tab" data-content="content2">Tab 2</div>
 <div class="tab" data-content="content3">Tab 3</div>
</div>

<div id="content1" class="content">
 <p>This is the content of Tab 1.</p>
</div>
<div id="content2" class="content">
 <p>This is the content of Tab 2.</p>
</div>
<div id="content3" class="content">
 <p>This is the content of Tab 3.</p>
</div>
```




```javascript
const tabs = document.getElementById('tabs');

// Event Delegation: Add a click event listener to the parent .tabs container
tabs.addEventListener('click', function(event) {
  // Check if the clicked element is a tab
  if (event.target.classList.contains('tab')) {
    const activeTab = document.querySelector('.tab.active');
    const activeContent = document.querySelector('.content.active');

    // Remove active classes from current active tab and content
    if (activeTab)
      activeTab.classList.remove('active');
    if (activeContent)
      activeContent.classList.remove('active');

    // Add active class to the clicked tab
    event.target.classList.add('active');

    // Show the associated content
    const contentId = event.target.getAttribute('data-content');
    const contentToShow = document.getElementById(contentId);
    contentToShow.classList.add('active');
  }
});
```

ΜΑΘΗΜΑ 2.6: Event Delegation

Πίνακες:

- Ένας μόνο listener (στο επίπεδο του container) ελέγχει τις αλληλεπιδράσεις με τον χρήστη

Παράδειγμα 4: tables.html

```
<table id="data-table">
 <thead>
 <tr>
  <th>Name</th>
  <th>Email</th>
  <th>Actions</th>
 </tr>
 </thead>
 <tbody>
 <tr>
  <td>John Doe</td>
  <td>john@example.com</td>
  <td>
   <span class="action-button" data-action="edit">Edit</span> |
   <span class="action-button" data-action="delete">Delete</span>
  </td>
 </tr>
 <tr>
  <td>Jane Smith</td>
  <td>jane@example.com</td>
  <td>
   <span class="action-button" data-action="edit">Edit</span> |
   <span class="action-button" data-action="delete">Delete</span>
  </td>
 </tr>
 </tbody>
</table>
```

Name	Email	Actions
John Doe	john@example.com	Edit | Delete
Jane Smith	jane@example.com	Edit | Delete

```
const table = document.getElementById('data-table');

// Event Delegation: Add event listener to the table
table.addEventListener('click', function(event) {
  // Check if the clicked element is an action button
  if (event.target.classList.contains('action-button')) {
    const action = event.target.getAttribute('data-action');
    const row = event.target.closest('tr');

    if (action === 'edit') {
      handleEdit(row);
    } else if (action === 'delete') {
      handleDelete(row);
    }
  }
});

function handleEdit(row) {
  const name = row.cells[0].textContent;
  const email = row.cells[1].textContent;
  console.log(`Editing: Name - ${name}, Email - ${email}`);
  // Additional code to edit the row can be implemented here
}
function handleDelete(row) {
  row.remove();
  console.log('Row deleted.');
}
```

Σημείωση:

- Η μέθοδος του target: closest(selector), επιστρέφει τον κοντινότερο πρόγονο στο στοιχείο - target που ικανοποιεί τον επιλογέα.