ΠΕΡΙΕΧΟΜΕΝΑ:

ΜΑΘΗΜΑ 2.1: ΔΙΑΔΟΣΗ EVENTS

1. Υπενθυμίσεις

JS Browser psounis YouTube

Υπενθύμιση από JS+Browser: Μάθημα 1.3:

- Στο μάθημα 1.3 είδαμε τις ακόλουθες μεθόδους που έχουν τα στοιχεία (επειδή στην αλυσίδα πρωτοτύπων τους έχουν το EventTarget)

Μέθοδος	Επεξήγηση
addEventListener (type, listener [, options])	Προσθέτει το event τύπου type. Όταν ενεργοποιείται, θα τρέχει ο κώδικας του listener.
removeEventListener (type, listener [, options])	Αφαιρεί το event τύπου type από τον κόμβο. Το αντικείμενο listener, είναι αυτό που προστέθηκε με την addEventListener

- Κατασκευάζουμε δικά μας event με τον κατασκευαστή:

Κατασκευαστής	Επεξήγηση
CustomEvent(name)	name: (Δικό μας) όνομα event (συμβολοσειρά)

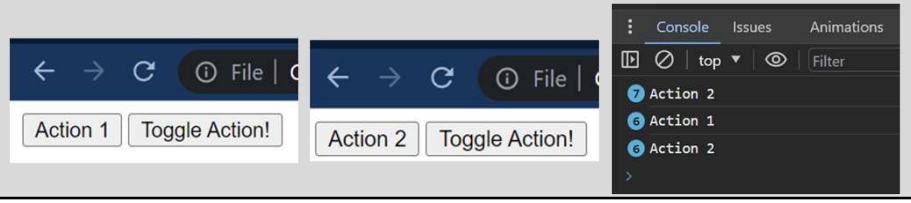
- ενώ στέλνουμε ένα event σε ένα στοιχείο, με τη μέθοδο του στοιχείου:

Μέθοδος	Επεξήγηση
dispatchEvent(event)	Στέλνει event στο αντικείμενο που την κάλεσε

Παράδειγμα 1: removeEventListener.html

```html
<body>
  <button>Click Me!</button>
  <button>Toggle Action!</button>
</body
```

```js
let listener1 = ()=>{
  console.log("Action 1");
};
let listener2 = ()=>{
  console.log("Action 2");
};
let button1 = document.querySelector("button:first-of-type");
let button2 = document.querySelector("button:last-of-type");

button1.addEventListener("click", listener1);
button1.textContent = "Action 1";

button2.addEventListener("click", ()=>{
  if (button1.textContent === "Action 1") {
    button1.removeEventListener("click", listener1);
    button1.addEventListener("click", listener2);
    button1.textContent = "Action 2";
  }
  else {
    button1.removeEventListener("click", listener2);
    button1.addEventListener("click", listener1);
    button1.textContent = "Action 1";
  }
});
```



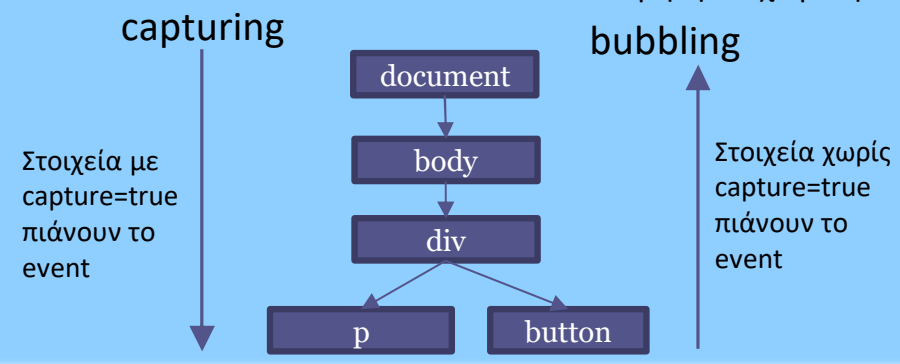
ΜΑΘΗΜΑ 2.1: ΔΙΑΔΟΣΗ EVENTS

2. Φάση Διάδοσης Events

Browser psounis

Ένα event από τη στιγμή που δημιουργείται (αυτόματα, π.χ. όταν γίνεται κλικ σε ένα στοιχείο) διατρέχει το document σε φάσεις:

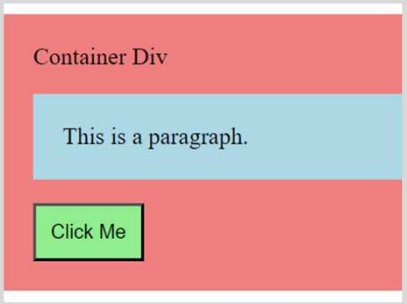
- 1η φάση (**event capturing**):
  - Κατασκευάζεται ένα μονοπάτι από τον κόμβο document προς τον κόμβο - στόχο του event
  - Αν κάποιος κόμβος του μονοπατιού έχει ορίσει listener με capture=true, πιάνει το event και μπορεί π.χ. να το σταματήσει (με την stopPropagation)
  - Έτσι τα γονικά στοιχεία έχουν την δυνατότητα να διαχειριστούν τα events πριν αυτά φτάσουν τον προορισμό τους.
- 2η φάση (**target**):
  - Το event φτάνει το στοιχείο-προορισμό του, όπου έγινε η δράση (π.χ. το κλικ)
  - Το event διαχειρίζεται από τους listeners που έχουν τεθεί στο στοιχείο
- 3η φάση (**event bubbling**):
  - Ακολουθείται το αντίστροφο μονοπάτι (από το στοιχείο προς το document)
  - Κόμβοι που έχουν ορίσει listener χωρίς το capture=true, μπορούν να πιάσουν το event και να κάνουν επιθυμητή διαχείριση



Παράδειγμα 2: event-phases.html

```html
<div class="container">
  Container Div
  <p class="text">This is a paragraph.</p>
  <button class="btn">Click Me</button>
</div>
```

```js
document.querySelector('.container').addEventListener('click', (event) => {
  console.log('Capturing: Container Div');
}, true); // true: capturing phase

document.querySelector('.text').addEventListener('click', (event) => {
  console.log('Target: Paragraph');
});

document.querySelector('.btn').addEventListener('click', (event) => {
  console.log('Target: Button');
});

document.querySelector('.container').addEventListener('click', (event) => {
  console.log('Bubbling: Container Div');
}); // default: false: bubbling phase
```




```
Capturing: Container Div
Target: Paragraph
Bubbling: Container Div
```

```
Capturing: Container Div
Target: Button
Bubbling: Container Div
```

ΜΑΘΗΜΑ 2.1: ΔΙΑΔΟΣΗ EVENTS

3.1. Έλεγχος Διάδοσης: stopPropagation()

Browser psounis

Έλεγχος διάδοσης:

- Προσφέρονται διαφορετικοί τρόποι για να κάνουμε έλεγχο της διάδοσης ενός event.
- Με την ακόλουθη μέθοδο του event:

Μέθοδος	Επεξήγηση
stopPropagation()	Σταματάει τη διάδοση του event (είτε στη φάση του capturing, είτε στη φάση του bubbling) Δεν αποτρέπει από το εκτελεστούν άλλοι handlers στο ίδιο στοιχείο Δεν αποτρέπει την default συμπεριφορά του στοιχείου

Παρατηρήσεις:

- Και ο στόχος του event (στο παράδειγμα τα buttons) περνάνε από φάση capturing και bubbling.
  - Αν και δεν συνηθίζεται να τις χρησιμοποιούμε, απαιτείται να γνωρίζουμε ότι υπάρχουν.
- Η default συμπεριφορά ποικίλει, ανάλογα με το στοιχείο:
  - Το <a> ανοίγει υπερσύνδεσμο
  - Το <input type="submit"> αποστέλλει τα δεδομένα φόρμας
  - κ.λπ.
  - Προσοχή, ότι η εμφάνιση ότι «πατιέται» το κουμπί, δεν είναι default συμπεριφορά, αλλά μέρος της γραφικής διεπαφής (CSS)

Παράδειγμα 3: stop-propagation.html

```html
<div class="container1">
  <button class="btn1">Click Me</button>
</div>
<div class="container2">
  <button class="btn2">Click Me</button>
</div>
```

```js
document.querySelector('.container1').addEventListener('click', (event) => {
  console.log('div: stop propagation');
  event.stopPropagation();
}, true); // true: capturing phase
document.querySelector('.btn2').addEventListener('click', (event) => {
  console.log('Target');
  event.stopPropagation();
});

document.querySelectorAll("button").forEach(button=>{
  button.addEventListener("click", ()=>{
    console.log("button capture")
  }, true)
  button.addEventListener("click", ()=>{
    console.log("button bubble")
  })
})
document.querySelectorAll("div").forEach(div=>{
  div.addEventListener("click", ()=>{
    console.log("div capture")
  }, true)
  div.addEventListener("click", ()=>{
    console.log("div bubble")
  })
})
```

```
div: stop propagation
div capture
div capture
button capture
Target
button bubble
```

Έλεγχος διάδοσης:

- Εναλλακτικά μπορούμε να χρησιμοποιήσουμε την ακόλουθη μέθοδο του event:

Μέθοδος	Επεξήγηση
stopImmediatePropagation()	Ίδια με τη stopPropagation() με τη διαφορά ότι σταματάει τους handlers στο στοιχείο που καλεί τη μέθοδο

Παράδειγμα 4: stop-immediate-propagation.html

```html
<div class="container1">
  <button class="btn1">Click Me</button>
</div>
<div class="container2">
  <button class="btn2">Click Me</button>
</div>
```

```js
document.querySelector('.container1').addEventListener('click', (event) => {
   console.log('div: stop propagation');
   event.stopImmediatePropagation();
}, true); // true: capturing phase
document.querySelector('.btn2').addEventListener('click', (event) => {
   console.log('Target');
   event.stopImmediatePropagation();
}); //  bubbling phase
document.querySelectorAll("button").forEach(button=>{
   button.addEventListener("click", ()=>{
      console.log("button capture")
   }, true)
   button.addEventListener("click", ()=>{
      console.log("button bubble")
   })
})
document.querySelectorAll("div").forEach(div=>{
   div.addEventListener("click", ()=>{
      console.log("div capture")
   }, true)
   div.addEventListener("click", ()=>{
      console.log("div bubble")
   })
})
```

```
div: stop propagation
div capture
button capture
Target
```

ΜΑΘΗΜΑ 2.1: ΔΙΑΔΟΣΗ EVENTS

3.3. Έλεγχος Διάδοσης: preventDefault()

Έλεγχος διάδοσης:

- Συμπληρωματικά, μπορούμε να χρησιμοποιήσουμε την ακόλουθη μέθοδο του event:

Μέθοδος	Επεξήγηση
preventDefault()	Αποτρέπει την εκτέλεση του default action του event

Σημείωση:

- Η μέθοδος μπορεί να κληθεί σε όλες τις φάσεις διάδοσης του event (το default action, ενεργοποιείται μετά τη διάδοση του event)

Σύνοψη:

- Οι μέθοδοι που χρησιμοποιούνται για τον έλεγχο διάδοσης του event και η χρησιμότητα τους

Μέθοδος	stopPropagation()	stopImmediate Propagation()	preventDefault()
Διακοπή φάσης capturing	Ναι	Ναι	Όχι
Διακοπή φάσης bubbling	Ναι	Ναι	Όχι
Σταματα άλλους handlers στο ίδιο στοιχείο	Όχι	Ναι	Όχι
Αποτρέπει την default ενέργεια	Όχι	Όχι	Ναι

Παράδειγμα 5: stop-immediate-propagation.html

```html
<div class="container1">
  <button class="btn1">Click Me</button>
</div>
<div class="container2">
  <button class="btn2">Click Me</button>
</div>
```

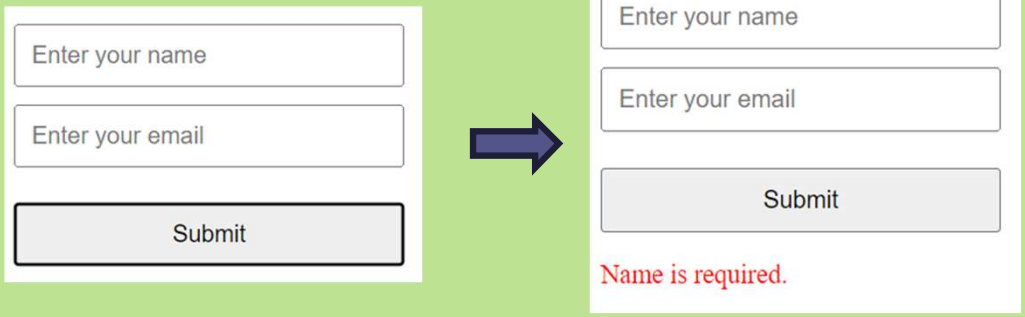
```js
document.querySelector('.container1').addEventListener('click', (event) => {
  console.log('div: stop propagation');
  event.stopPropagation();
}, true); // true: capturing phase
document.querySelector('.btn2').addEventListener('click', (event) => {
  console.log('Target');
  event.stopPropagation();
});

document.querySelectorAll("button").forEach(button=>{
  button.addEventListener("click", ()=>{
    console.log("button capture")
  }, true)
  button.addEventListener("click", ()=>{
    console.log("button bubble")
  })
})
document.querySelectorAll("div").forEach(div=>{
  div.addEventListener("click", ()=>{
    console.log("div capture")
  }, true)
  div.addEventListener("click", ()=>{
    console.log("div bubble")
  })
})
```

```
div: prevent default
div capture
div bubble
div capture
Target
div bubble
```

4. Παράδειγμα: Form Validation

Συχνή Χρήση:

- Front-End Validation σε Φόρμες:
  - Σε φόρμες, μπορούμε να έχουμε ένα event που ενεργοποιείται όταν ο χρήστης κάνει κλικ στο κουμπί submit στο επίπεδο της φόρμας
  - Αν υπάρχουν λάθη στη συμπλήρωση, τότε ο handler κάνει preventDefault() και προβάλλει κάποιο μήνυμα λάθους



Σημείωση:

- Είναι χρήσιμο ότι έχουμε τον έλεγχο στο επίπεδο της φόρμας, γιατί έτσι πιάνουμε όχι μόνο την περίπτωση που ο χρήστης κάνει κλικ στο «submit», αλλά και άλλες περιπτώσεις, όπως να πατηθεί το Enter, να γίνει προγραμματιστική υποβολή της φόρμας μέσω JS κ.α.

Παράδειγμα 6: form-validation

```html
<form class="form-container" id="myForm">
  <input type="text" id="name" class="input-field" placeholder="Enter your name">
  <input type="email" id="email" class="input-field" placeholder="Enter your email">
  <button type="submit">Submit</button>
  <div id="error-message" class="error"></div>
</form>
```

```javascript
document.getElementById('myForm').addEventListener('submit', function(event) {
    const name = document.getElementById('name');
    const email = document.getElementById('email');
    const errorMessage = document.getElementById('error-message');

    errorMessage.textContent = '';

    let isValid = true;
    if (name.value.trim() === '') {
        isValid = false;
        errorMessage.textContent = 'Name is required.';
    } else if (email.value.trim() === '') {
        isValid = false;
        errorMessage.textContent = 'Valid email is required.';
    }

    if (!isValid) {
        event.preventDefault(); // Prevent the default form submission
    }
});
```