

WMNFE 2210

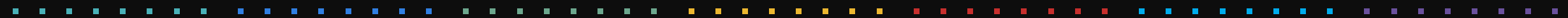
FRONT-END DEVELOPMENT

JavaScript #6 | Events





ΠΕΡΙΕΧΟΜΕΝΑ



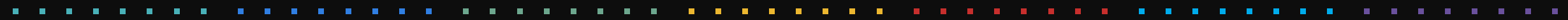


Περιεχόμενα

- Accessing DOM Elements
- Browser events
 - Mouse events
 - Key events
 - Form elements events
 - Document events
- Event registration
 - **this** & **event.target**
- Event methods



ACCESSING DOM ELEMENTS





Accessing DOM Elements

getElementById()

Η μέθοδος **getElementById()** επιστρέφει ένα αντικείμενο που αντιστοιχεί στο στοιχείο εκείνο με το δεδομένο **id**.

Μιας και τα **ids** των στοιχείων οφείλουν να είναι μοναδικά, είναι ένα γρήγορος τρόπος να αποκτήσουμε πρόσβαση σε συγκεκριμένο στοιχείο του **DOM**.

```
let element = document.getElementById(id);
```



Accessing DOM Elements

getElementsByClassName()

Η μέθοδος **getElementsByClassName()** επιστρέφει μια "live" συλλογή με τα στοιχεία απογόνους (χωρίς τον γονέα) που περιέχουν το συγκεκριμένο όνομα (ή ονόματα) κλάσης.

Η μέθοδος αυτή μπορεί να χρησιμοποιηθεί είτε στο αντικείμενο **document**, είτε σε οποιοδήποτε άλλο αντικείμενο.

```
let elements = element.getElementsByClassName (names);
```



Accessing DOM Elements

getElementsByTagName()

Η μέθοδος **getElementsByTagName()** επιστρέφει μια "live" συλλογή με τα στοιχεία απογόνους (χωρίς τον γονέα) που ανήκουν στη συγκεκριμένη ετικέτα (ή ετικέτες).

Η μέθοδος αυτή μπορεί να χρησιμοποιηθεί είτε στο αντικείμενο **document**, είτε σε οποιοδήποτε άλλο αντικείμενο.

```
let elements = element.getElementsByTagName(tagNames);
```



Accessing DOM Elements

querySelector()

Η μέθοδος **querySelector()** επιστρέφει το πρώτο στοιχείο απόγονο που ικανοποιεί τον συγκεκριμένο επιλογή (ή επιλογείς).

Η μέθοδος αυτή μπορεί να χρησιμοποιηθεί είτε στο αντικείμενο **document**, είτε σε οποιοδήποτε άλλο αντικείμενο.

```
let element = baseElement.querySelector(selectors);
```




Accessing DOM Elements

querySelectorAll()

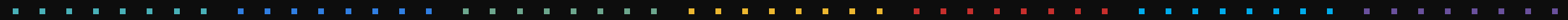
Η μέθοδος **querySelector()** επιστρέφει μια συλλογή (όχι live) με τα στοιχεία απογόνους που ικανοποιούν τον συγκεκριμένο επιλογή (ή επιλογείς).

Η μέθοδος αυτή μπορεί να χρησιμοποιηθεί είτε στο αντικείμενο **document**, είτε σε οποιοδήποτε άλλο αντικείμενο.

```
let list = parentNode.querySelectorAll(selectors);
```



BROWSER EVENTS



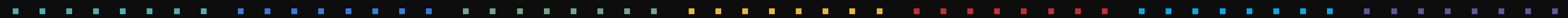


Browser Events

Γενικά

Ένα γεγονός (**event**) είναι ένας τρόπος «ειδοποίησης» ότι κάτι «ενδιαφέρον» έχει συμβεί στη σελίδα. Όλα τα στοιχεία μπορούν να παράξουν γεγονότα, αλλά σε μία τυπική σελίδα κάποια συγκεκριμένα γεγονότα είναι και τα πιο συνηθισμένα.

Κάθε γεγονός αναπαριστάται από ένα **event object**, το οποίο περιέχει πληροφορίες για αυτό.





Browser Events

Τυπικά γεγονότα

- Όταν φορτώσει η σελίδα
- Όταν ο χρήστης κάνει κλικ με το mouse
- Όταν φορτώσει μια εικόνα
- Όταν το mouse βρεθεί πάνω από ένα στοιχείο
- Όταν ο χρήστης πληκτρολογήσει κάτι σε ένα text field
- Όταν υποβληθεί μια φόρμα



Browser Events

Mouse events

- **click** – όταν κάνουμε κλικ
- **dblclick** – όταν κάνουμε διπλό κλικ
- **mousedown** – όταν ένα κουμπί από το ποντίκι πατηθεί
- **mouseover** – όταν το ποντίκι βρεθεί πάνω από στοιχείο
- **mouseout** – όταν το ποντίκι φύγει πάνω από στοιχείο
- **mousemove** – όταν κινείται το ποντίκι



Browser Events

Keyboard events

- **keydown**
όταν πατηθεί ένα πλήκτρο
- **keyup**
όταν αφεθεί ένα πλήκτρο



Browser Events

Form elements events

- **change**
όταν ένα στοιχείο αλλάξει
- **focus**
όταν ένα στοιχείο αποκτήσει την εστίαση
- **blur**
όταν ένα στοιχείο χάσει την εστίαση
- **submit**
όταν υποβληθεί μία φόρμα



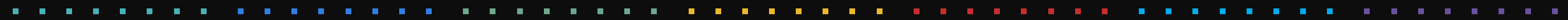
Browser Events

Document events

- **DOMContentLoaded**
όταν η **HTML** έχει φορτωθεί και αναλυθεί πλήρως
- **load**
όταν η σελίδα (μαζί με όλους τους πόρους της) ή ένα στοιχείο (όπως μια εικόνα) φορτωθεί
- **unload**
όταν η σελίδα έχει «εκφορτωθεί» (ή το παράθυρο του προγράμματος περιήγησης έχει κλείσει)



EVENT REGISTRATION





Event registration

HTML-attribute

```
<body onload="init()">
  <script>
    function init() {
      console.log('Page loaded');
    }

    function click() {
      console.log('Button clicked');
    }
  </script>
  <button onclick="click()">Click me</button>
</body>
```



Event registration

DOM property

```
<body>
  <button id="click-me">Click me</button>
  <script>
    function init() {
      console.log('Page loaded');
    }

    window.onload = init;

    function click() {
      console.log('Button clicked');
    }

    document.getElementById('click-me').onclick = click;
  </script>
</body>
```



Event registration

addEventListener

```
<body>
  <button id="click-me">Click me</button>
  <script>
    function init() {
      console.log('Page loaded');
    }

    window.addEventListener('load', init);

    function click() {
      console.log('Button clicked');
    }

    document.getElementById('click-me').addEventListener('click', click);
  </script>
</body>
```



Event registration

1337 h4x0r way

```
//external script

window.addEventListener('load', () => {
  console.log('Page loaded');
});

document.getElementById('click-me').addEventListener('click', () => {
  console.log('Button clicked');
});

// better yet, in most cases
document.addEventListener('DOMContentLoaded', () => {
  console.log('DOM content loaded');
});
```



Event registration

`document.getElementById('id')` gives **null**

Ένα συνηθισμένο πρόβλημα, όταν «αναζητούμε» κάποιο στοιχείο του DOM, είναι το εξής:

Η μέθοδος που καλούμε επιστρέφει **null**, όχι επειδή η κλήση μας ήταν λανθασμένη, αλλά επειδή το στοιχείο δεν έχει δημιουργηθεί ακόμα. Αυτό συμβαίνει όταν το συγκεκριμένο κομμάτι κώδικα εκτελείται πριν την ανάλυση (**parsing**) της **HTML**, που περιέχει το εν λόγω στοιχείο.



Event registration

`document.getElementById('id')` gives **null**

Για να αντιμετωπιστεί το συγκεκριμένο ζήτημα μπορούμε να φροντίσουμε για ένα από τα παρακάτω:

- Ο κώδικάς μας να εκτελείται στο **load** ή στο **DOMContentLoaded event**.
- να ενσωματώσουμε το συγκεκριμένο **script** στο τέλος της ετικέτας **body**.
- να χρησιμοποιήσουμε το **attribute defer** της ετικέτας **script**.



Event registration

this keyword

Με τη λέξη κλειδί **this** μπορούμε, γενικά, να αναφερθούμε στο αντικείμενο «ιδιοκτήτη» (**owner object**).

Στα πλαίσια μιας **function**, η οποία χειρίζεται ένα γεγονός (**event handler/listener**), η λέξη κλειδί **this** αναφέρεται στο στοιχείο στο οποίο αφορά το συγκεκριμένο γεγονός.

Προσοχή, όταν χρησιμοποιούμε τη λέξη κλειδί **this** με **arrow functions**!

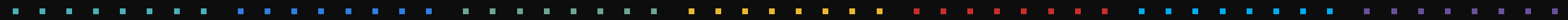


Event registration

Event & Event.target

Μία καλύτερη πρακτική, για να αντλήσουμε πληροφορίες για ένα γεγονός, αλλά και για το αντικείμενο στο οποίο αφορά, είναι να κάνουμε χρήση της παραμέτρου τύπου **Event**.

Το αντικείμενο αυτό "περιέχει" πολύ χρήσιμες πληροφορίες για το γεγονός, όπως το στοιχείο στο οποίο αφορά το γεγονός αυτό.





Event registration

this / event.target

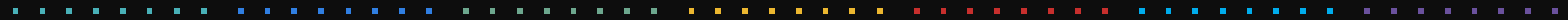
```
const el1 = document.getElementById('el1');
const el2 = document.getElementById('el2');

el1.addEventListener('mouseover', function (event) {
  console.log('this', this);
  console.log('event', event);
  console.log('event.target', event.target);
});

el2.addEventListener('mouseover', (event) => {
  console.log('this', this); // Oops
  console.log('event', event);
  console.log('event.target', event.target);
});
```



EVENT METHODS





Event methods

preventDefault()

Η μέθοδος **preventDefault()** σταματά την default λειτουργία ενός συμβάντος, για παράδειγμα την αλλαγή σελίδας στο **click** μιας ετικέτας **a** ή στο **submit** μιας φόρμας.

```
event.preventDefault();
```



Event methods

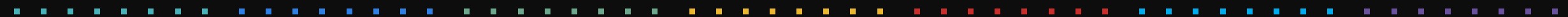
stopPropagation()

Η μέθοδος **stopPropagation()** σταματά την περαιτέρω διάδοση ενός συμβάντος, για παράδειγμα το **click** εντός στοιχείου στο γονέα του.

```
event.stopPropagation();
```



HOMework





Homework

Countdown timer

Επισκεφτείτε το αποθετήριο <https://github.com/nbilalis/countdown-timer> και «κλωνοποιήστε» το.


Ακολουθώντας τις οδηγίες, που βρίσκονται στο **README.md** αλλά και στο αρχείο **index.html**, προσπαθήστε να ολοκληρώσετε τον *timer* ώστε να είναι λειτουργικός.



Χρήσιμα links

 Event reference | MDN

<https://developer.mozilla.org/en-US/docs/Web/Events>

 Browser: Document, Events, Interfaces

<https://javascript.info/ui>

 EventTarget.addEventListener() - Web APIs

[https://developer.mozilla.org/en-US/docs/Web/API/Eve...](https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener)

 HTML DOM addEventListener() Method

https://www.w3schools.com/jsref/met_element_addev...



Extra info

 Can I use... Support tables for HTML5, CSS3, etc
<https://caniuse.com/?search=addEventListener>

 Window vs. Document Loading Events
<https://gist.github.com/jsonberry/0d71007ea188785e1a...>



THANK YOU!