ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

Client-side (Frontend) JavaScript & HTML-DOM

Αθανάσιος Ανδρούτσος

Web Browsers (1)

- Στο κεφάλαιο αυτό θα διασυνδέσουμε τις τρεις βασικές τεχνολογίες στο Web
  - **HTML** / **CSS** και **JavaScript**
- Οι browsers είναι **HTML renders** και μπορούν και παρουσιάζουν formatted text και images αλλά από την άλλη πλευρά είναι και μία **πλατφόρμα (σαν ένα λειτουργικό σύστημα) για application development** παρέχοντας services, όπως networking, storage και threading

WebAssembly (1)

- Ιστορικά η βασική γλώσσα προγραμματισμού στους Browsers ήταν η **JavaScript**
- Πρόσφατα (2015) υποστηρίζεται και μία 2η γλώσσα η **WebAssembly** (WASM) που η ιδέα είναι να μπορεί να προσφέρει near-native speeds σε performance-sensitive εφαρμογές όπως 3D Graphics, Virtual and Augmented Reality Apps, Encryption, Music Applications, Image recognition, Language Interpreters, Visualizations, κλπ.

WebAssembly (2)

- Σκοπός δεν είναι να γράφουμε με το χέρι WASM αλλά αρκετές γλώσσες όπως C/C++, C# (ASP.NET Blazor), και ιδιαίτερα η Rust κάνουν transpiling σε WASM δίνοντας έτσι τη δυνατότητα σε αυτές τις γλώσσες να τρέξουν στον Browser με πολύ γρήγορες ταχύτητες
- Μπορεί επίσης η WASM να λειτουργήσει συμπληρωματικά με την JavaScript για critical performance εφαρμογές

Web Browsers (1)

- Στο παρόν κεφάλαιο θα ασχοληθούμε με την **JavaScript που τρέχει στους Web Browsers** (JavaScript run in web browsers) που είναι συνώνυμο με το client-side JavaScript σε αντίθεση με το server-side JavaScript που είναι η JavaScript (Node.js) που τρέχει σε Web Servers
- Τα client-side και server-side είναι το frontend και backend αντίστοιχα στις Client-Server και N-Tier αρχιτεκτονικές

Web Browsers (2)

- Η JavaScript μπορεί να χρησιμοποιεί τις δυνατότητες των Web Browsers να εμφανίζουν περιεχόμενο στις HTML σελίδες
- Μπορεί επίσης και να αξιοποιεί τα networking, storage, threading services της πλατφόρμας που παρέχουν οι Web Browsers

Web Browsers (3)

- Η ιδέα είναι πως οι Web Browser Vendors παρέχουν ένα Web API που υλοποιείται από την JavaScript (είναι υλοποιημένο σε C++)
- Η JavaScript επομένως μας παρέχει ένα JavaScript Standard Library για να αλληλεπιδρούμε με τον Browser
- Το Mozilla MDN web docs project (https://developer.mozilla.org/) μας δίνει πληροφορίες (documentation) για αυτό το Web API

Web Browsers (4)

- Ιστορικά κάποια στοιχεία των Web APIs έχουν επικρατήσει (standardized) και άλλα έχουν γίνει obsolete
- Για παράδειγμα το *innerHTML* property έχει γίνει standard
- Άλλα όπως το *document.write()* δεν θεωρούνται πλέον αποδεκτά

HTML/CSS/JavaScript

- Τα Web APIs εξελίχθηκαν και όλοι πλέον οι Web Browsers έχουν υιοθετήσει ένα εξελιγμένο Web API, που παρέχεται στο πλαίσιο του **DOM (Document Object Model)**
- Το **DOM (Document Object Model)** είναι μία δενδρική αναπαράσταση *στη μνήμη* ενός HTML αρχείου όπου **κάθε *στοιχείο HTML*** αναπαρίσταται ως ένας κόμβος του δένδρου *με ιδιότητες και* ***μεθόδους (public API)*** που επιτρέπουν την προγραμματιστική πρόσβαση στα στοιχεία HTML
- **Αυτό που βλέπουμε σε ένα browser, είναι η οπτική αναπαράσταση του DOM**

HTML DOM (2)

- Το DOM αποτελείται από κόμβους (Nodes) και ο κάθε κόμβος συνδέεται με κάποιο ή κάποιους child nodes, εκτός από τα leaf nodes (τα φύλλα του δένδρου, οι τελικοί κόμβοι) που δεν έχουν children
- Το DOM έχει επομένως μία ιεραρχία δένδρου (tree). Κάθε κόμβος του δένδρου αναπαριστά κατά βάση ένα **HTML Element**, ενώ υπάρχουν και άλλοι τύποι nodes όπως θα δούμε όπως Text Nodes, Attribute Nodes

HTML DOM (2)

Interface	nodeType constant	nodeType value
Element	Node.ELEMENT_NODE	1
Text	Node.TEXT_NODE	3
Document	Node.DOCUMENT_NODE	9
Comment	Node.COMMENT_NODE	8
DocumentFragment	Node.DOCUMENT_FRAGMENT_NODE	11
Attr	Node.ATTRIBUTE_NODE	2

- Αριστερά αναφέρονται οι πιο γνωστοί τύποι κόμβων

DOM Specs

- Κάθε κόμβος του DOM αναπαρίσταται από ένα interface που παρέχει ένα public API που μπορεί να υλοποιηθεί
- Γενικά το DOM είναι specs (ιεραρχία από interfaces) που κάθε γλώσσα μπορεί να υλοποιήσει και να δώσει η γλώσσα ένα API προς τους προγραμματιστές

OMG IDL (1)


```
interface Node {
  // NodeType
  const unsigned short      ELEMENT_NODE       = 1;
  const unsigned short      ATTRIBUTE_NODE     = 2;
  const unsigned short      TEXT_NODE          = 3;
  const unsigned short      CDATA_SECTION_NODE = 4;
  const unsigned short      ENTITY_REFERENCE_NODE = 5;
  const unsigned short      ENTITY_NODE        = 6;
  const unsigned short      PROCESSING_INSTRUCTION_NODE = 7;
  const unsigned short      COMMENT_NODE       = 8;
  const unsigned short      DOCUMENT_NODE      = 9;
  const unsigned short      DOCUMENT_TYPE_NODE = 10;
  const unsigned short      DOCUMENT_FRAGMENT_NODE = 11;
  const unsigned short      NOTATION_NODE      = 12;
```

- Τα specs περιγράφονται σε CORBA IDL (Interface Definition Language) και υλοποιούνται από την JavaScript
- https://www.w3.org/TR/REC-DOM-Level-1/idl-definitions.html

OMG IDL (2)

- Στο IDL ορίζεται μία ιεραρχία κόμβων. Για παράδειγμα το **HTMLElement** κληρονομεί από το **Element**, που με τη σειρά του κάνει extends το **Node**
- Οι πιο σημαντικοί τύποι κόμβων (nodes) του DOM σχετίζονται με **HTML Elements** (interface HTMLElement), **ιδιότητες** ενός HTML στοιχείου (interface ***Attr***) ή το **περιεχόμενο** ενός HTML στοιχείου (interface ***Text***)

Document object (1)


```
interface Document : Node {
  readonly attribute  DocumentType         doctype;
  readonly attribute  DOMImplementation    implementation;
  readonly attribute  Element              documentElement;
  Element                    createElement(in DOMString tagName)
                                          raises(DOMException);
  DocumentFragment           createDocumentFragment();
  Text                       createTextNode(in DOMString data);
  Comment                    createComment(in DOMString data);
  CDATASection               createCDATASection(in DOMString data)
                                               raises(DOMException);
  ProcessingInstruction      createProcessingInstruction(in DOMString target,
                                                         in DOMString data)
                                                         raises(DOMException);
  Attr                       createAttribute(in DOMString name)
                                            raises(DOMException);
  EntityReference            createEntityReference(in DOMString name)
                                                  raises(DOMException);
  NodeList                   getElementsByTagName(in DOMString tagname);
};
```

- Η JavaScript υλοποιεί τα DOM Specs με JS objects. Για παράδειγμα το ***document*** *object της JS* ***υλοποιεί*** *το* Document Node, που αναπαριστά το HTML document

Document object (1)

- Το **JavaScript Global Object** στους Web Browsers μας παρέχει μία ιδιότητα **window** που αναπαριστά το window (tab) που τρέχει η JavaScript
- Το Global Object μας παρέχει επίσης την ιδιότητα **document** που αναπαριστά όλο το HTML document
- Το document object περιέχει τις ιδιότητες **body** και **head** που αναφέρονται στα Element objects για τα <body> και <head> html tags
- Αν θέλουμε ωστόσο να αναφερθούμε σε άλλα στοιχεία του DOM **θα πρέπει να μπορούμε να τα εντοπίσουμε και να τα επιλέξουμε**

Document object (2)

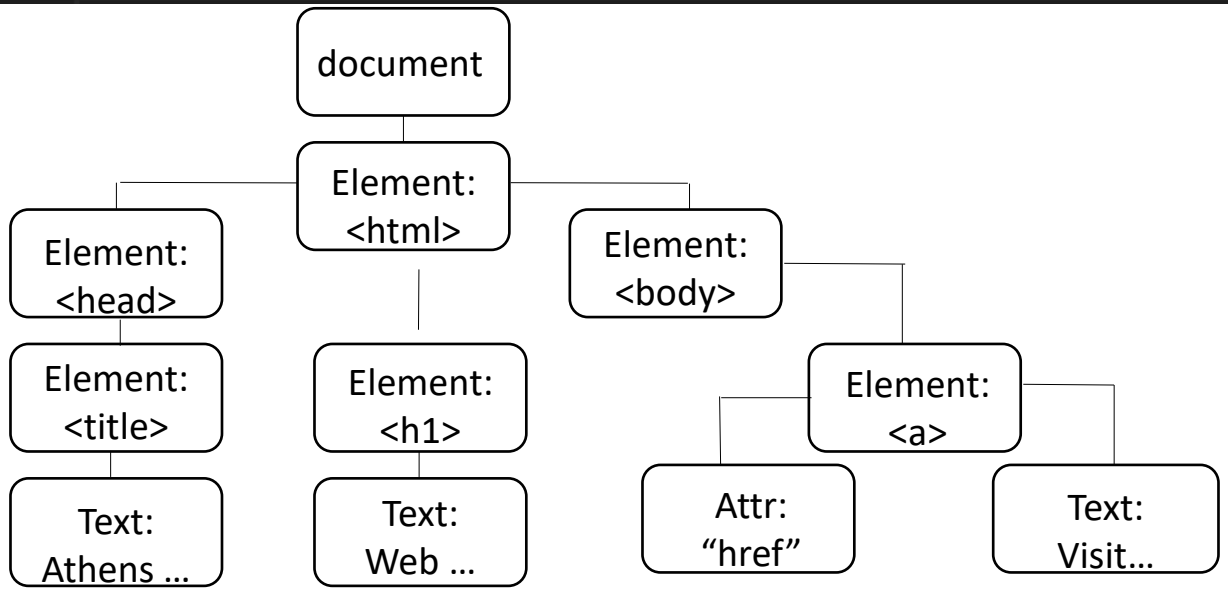
- Το global object (**window** / **globalThis**) αντιστοιχεί στο browser window και είναι το object όπου ορίζεται η standard library της JavaScript
- Πιο ψηλά στην ιεραρχία ενός HTML page βρίσκεται το **window object** που αναπαριστά το παράθυρο του browser. Πιο κάτω βρίσκεται το ***document*** object που αναπαριστά τη σελίδα. Πιο κάτω βρίσκεται το <*html*> και μετά τα **head** και **body** objects
- Και τελικά πιο κάτω όλα τα HTML Elements και λοιπά στοιχεία μίας Web Σελίδας

DOM (Document Object Model)

HTML DOM

testbed > dom.html > ...

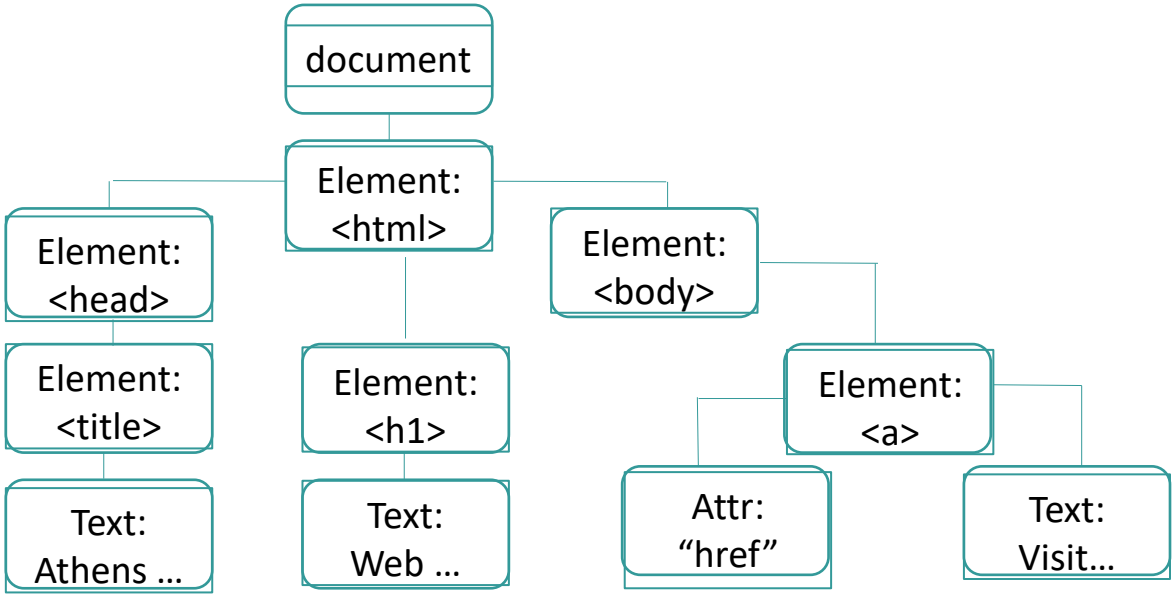
```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <title>Athens University of Economics and Business</title>
5   </head>
6   <body>
7       <h1>Web Development Course</h1>
8       <a href="https://elearning.aueb.gr/">Visit AUEB eLearning</a>
9   </body>
10  </html>
```



- Αριστερά-κάτω αναπαρίσταται το DOM της αντίστοιχης σελίδας που περιλαμβάνει το document object καθώς και άλλους κόμβους που μπορεί να είναι Element nodes, Attribute nodes, Text nodes

DOM - Σχέσεις κόμβων

HTML DOM



- Ο **root node** του document είναι το **document** object
- Κάτω από το root node είναι το <html> element
- Κάθε κόμβος έχει ένα γονέα (**parent**) καθώς και προγόνους. Ένας κόμβος μπορεί να έχει απογόνους (**children**)
- Οι κόμβοι του ίδιου επιπέδου κάτω από έναν κοινό άμεσο πρόγονο είναι **siblings**

- Το <head> (document.head) είναι το first child node του html node και το <body> (document.body) το last child node του <html> node
- Το <head> έχει ένα child, το <title> που είναι Element node
- Το <title> έχει ένα child το "Athens …" που είναι Text node
- Το ίδιο και το <h1> έχει ένα child το "Web … " που είναι Text node
- Το <a> έχει ένα sibling το <h1> και δύο children το "href" που είναι Attribute node και ένα Text node, το "Visit AUEB eLearning"

Δημιουργία DOM

- Το DOM μπορεί να σχηματιστεί είτε από στατική σελίδα HTML ή δυναμικά μόνο με JavaScript
- Δεν είναι απαραίτητο δηλαδή να σχηματιστεί από στατική σελίδα HTML
- Μπορεί δηλαδή η JavaScript να δημιουργεί δυναμικά όλα το DOM

JavaScript και Web Σελίδες (1)

- Η JavaScript μπορεί να δημιουργήσει το DOM, καθώς και να **αλληλεπιδράσει τόσο με τα HTML στοιχεία όσο και με τα CSS στοιχεία του DOM**
- Η HTML σελίδα μπορεί να είναι στατική και η JS να αλληλεπιδρά μέσω του DOM ή απλά να παρέχει ένα root div και όλο το άλλο περιεχόμενο να έρχεται δυναμικά από τη JavaScript
- Στην απλή Vanilla JavaScript το script tag μέσα στο HTML αρχείο διασυνδέει το HTML αρχείο με το JS Script, όπου με την ιδιότητα src μπορούμε να φορτώσουμε ένα εξωτερικό αρχείο με κατάληξη .js

JavaScript και Web Σελίδες (2)

- Μπορούμε μέσω της JavaScript να:
  - **προσθέσουμε**, **τροποποιήσουμε** ή να **διαγράψουμε** HTML στοιχεία και ιδιότητες του DOM
  - να τροποποιήσουμε CSS ιδιότητες των κόμβων του DOM
  - να δημιουργήσουμε και διαχειριστούμε events

JavaScript και Web Σελίδες (3)

- Θυμίζουμε και υπογραμμίζουμε ότι αυτό που κάνει render ο Browser είναι το DOM. Αυτό που βλέπουμε είναι το DOM, δηλαδή η οπτική αναπαράσταση του DOM. **Οι αλλαγές στο DOM άμεσα εμφανίζονται στην οθόνη.**
- Για να μπορέσει να γίνει η αλληλεπίδραση με το DOM, η JavaScript μας παρέχει μία **βιβλιοθήκη συναρτήσεων/μεθόδων δηλαδή ένα API** έτσι ώστε να μπορούμε:
  - Να **επιλέγουμε** στοιχεία HTML
  - Να τα επεξεργαζόμαστε (**δημιουργούμε**, **τροποποιούμε**, **διαγράφουμε**)

Επιλογή στοιχείων του DOM

- Οι βασικοί μηχανισμοί επιλογής στοιχείων του DOM που παρέχονται από την JS είναι:
  - **document.querySelector("CSS Selector")** που επιλέγει στοιχεία απογόνους του document με βάση ένα CSS selector. Επιστρέφει ένα δείκτη στο 1ο Element που ταιριάζει στην επιλογή ή null αν δεν βρεθεί**. Παρέχεται επίσης και ως μέθοδος του Element** (βλ. επόμενες διαφάνειες)
  - **document.getElementByID("id")** που επιλέγει στοιχεία με βάση το id του στοιχείου. Επιστρέφει ένα δείκτη στο Element που ταιριάζει στην επιλογή ή null αν δεν βρεθεί. Το getElementByID είναι μέθοδος μόνο του document object

querySelectorAll() (1)

- Η **querySelectorAll()** είναι παρόμοια με την querySelector() αλλά επιστρέφει όλα τα στοιχεία, όχι μόνο το πρώτο. Για παράδειγμα:
  - document.querySelectorAll('h1')
  - document.querySelectorAll('h1, h2')
- Επιστρέφεται **NodeList** που είναι δομή παρόμοια με Array (array-like object). Έχει length και μπορούμε να διατρέξουμε την επιστρεφόμενη λίστα με for, for/of ή .forEach(). Μπορούμε να μετατρέψουμε σε array, με Array.from()
- Αν δεν υπάρχουν/επιστραφούν στοιχεία, το length είναι 0

querySelectorAll() (2)

- Οι querySelector() και querySelectorAll() ορίζονται στο **Document αλλά και στο Element**
- Αν κληθούν σε κάποιο element επιστρέφουν αποτελέσματα που είναι απόγονοι του Element
- Για παράδειγμα myPar.querySelectorAll('a') που επιλέγει όλα τα <a> που είναι απόγονοι της παραγράφου myP

getElementBy* (1)

- Πιο παλιός (legacy) τρόπος επιλογής στοιχείων, που είναι πλέον obsolete:
- **getElementById('id')** επιλέγει μοναδικό στοιχείο με βάση το id ή null (querySelector('#id'))
- **getElementsByName**('product') επιλέγει στοιχεία με βάση την ιδιότητα name (querySelectorAll('[name="product"]'))
- **getElementsByTagName**('p') επιλέγει με βάση το tag (querySelectorAll('p'))
- **getElementsByClassName**('myP') επιλέγει με βάση το class name (querySelectorAll('.myP'))

getElementBy* (2)

- Οι getElementsBy* μέθοδοι επιστρέφουν '**live**' NodeList
- Το περιεχόμενο και το length αλλάζει αυτόματα, όταν αλλάξει το DOM

Δομή DOM και traversal

- Όταν έχουμε επιλέξει ένα Element του Document, θα ήταν πολλές φορές επιθυμητό να μπορούμε να ‘βρούμε’ και να επιλέξουμε related portions του Element (**parent, children, siblings**)
- Αν ενδιαφερόμαστε για related Elements τότε κάθε Element object μας παρέχει αντίστοιχα properties για να αναφερθούμε στα parent, siblings και children objects

Related Elements

- **Ιδιότητες (Properties) του Element object**
  - ***parentNode*** – αναφέρεται στον parentNode
  - ***children*** – επιστρέφει NodeList με τα element nodes που είναι παιδιά ενός κόμβου αλλά κάνει exclude τα Text nodes (και comments nodes). Με children[nth] επιλέγουμε το nth στοιχείο – το 0 είναι το 1ο index
  - ***firstElementChild / lastElementChild*** – παρόμοια με children[0] και children[length – 1]
  - ***nextElementSibling / previousElementSibling*** – επόμενο Element Node, προηγούμενο Element Node αντίστοιχα
- Για παράδειγμα: document.children[0].children[0] // <head> (ή κατευθείαν document.head)

Preorder traversal (1)


```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, ini
7       <title>Traverse DOM</title>
8   </head>
9   <body>
10      <h1>Coding Factory</h1>
11      <p>The best coding school!</p>
12      <a href="https://codingfactory.aueb.gr/">Visit is</a>
13
14      <script src="./js/traverseDOM.js"></script>
15  </body>
16  </html>
```

Preorder traversal (2)


```
1   /**
2    * Visits each node in a pre-order mode.
3    * That is root, left node, right node.
4    *
5    * @param {Node} tree - the tree root node
6    */
7   function preOrderTraversal(tree) {
8       console.log(tree)
9       for (const subtree of tree.children) {
10          preOrderTraversal(subtree)
11      }
12  }
13
14  preOrderTraversal(document)
```

- Κάνουμε Depth First Search (DFS) και συγκεκριμένα Preorder Traversal - Προδιατεταγμένη διάσχιση (Ρίζα-Αριστερά-Δεξιά)
- Εκτυπώνουμε όλο το tree εκτός από text nodes που δεν επιστρέφονται από την property children
- Η preOrderTraversal είναι αναδρομική. Εκτυπώνει υπόδενδρα αναδρομικά. Εδώ ξεκινάμε από το document

Related Nodes

- Αν θέλουμε τα related nodes (συμπεριλαμβανομένων text Nodes και Comment Nodes και όχι μόνο related Elements ενός Element) μας παρέχονται ιδιότητες του **Node object**
  - ***parentNode*** – αναφέρεται στον parentNode όποιος κι αν είναι
  - ***childNodes*** – NodeList με όλα τα child nodes ανεξάρτητα από το αν είναι elements ή όχι. Με childNodes[nth] επιλέγουμε το nth στοιχείο – το 0 είναι το 1ο index
  - ***firstChild*** / ***lastChild*** – παρόμοια με childNodes[0] και childNodes[length – 1]
  - ***nextSibling*** / ***previousSibling*** – επόμενος κόμβος Node, προηγούμενος κόμβος Node αντίστοιχα
  - nodeType / nodeValue / nodeName – nodeType: 1 Element Nodes, 3 Text Nodes , 9 Document Node, nodeVaue: textual content για Text και Comment nodes, nodeName: tag name (σε UpperCase) αν είναι Element

Traverse Text nodes


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Hello</h1>
    <p>World</p>

    <script>
        function traverseText(t) {
            let s = ""
            for (let child = t.firstChild; child !== null; child = child.nextSibling) {
                if (child.nodeType === 3) s += child.nodeValue + " "
                else if (child.nodeType === 1 && child.nodeName !== 'SCRIPT') {
                    s +=  traverseText(child)

                }
            }

            return s
        }

        console.log(traverseText(document.body))
    </script>
</body>
</html>
```

- Traverse tree, and select Text Nodes (type === 3)
- Αναδρομικά αν το στοιχείο είναι Element και διάφορο από το ‘SCRIPT’ tag

HTML Attributes (1)

- Κάθε node στο DOM αντιστοιχεί σε ένα JavaScript object. Κάθε τέτοιο object περιλαμβάνει properties που αντιστοιχούν στα HTML attributes του συγκεκριμένου HTML Element
- Το γενικό Element object αντιστοιχεί properties σε attributes που έχουν όλα τα HTML elements, όπως **id, style, lang, title, onclick** (για click event)
- Τα πιο συγκεκριμένα Element objects ορίζουν τις δικές τους ιδιότητες. Για παράδειγμα, το HTMLImageElement που αναπαριστά ένα <img> ορίζει τις ιδιότητες src, alt, width, height, κλπ.

HTML Attributes (2)

- Μπορούμε δυναμικά να δημιουργήσουμε ένα HTMLImageElement ως const imgElement = new HTMLImageElement();
- Ωστόσο είναι πιο συνηθισμένο και απλό να δημιουργούμε elements με βάση το html tag τους ως
- const imgElement = document.createElement('img');
- imgElement.src = ‘./cf.jpg';
- imgElement.alt = Coding Factory AUEB';

Create Element and Append

testbed > createElement.html > html > body > script

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-
6      <title>Document</title>
7  </head>
8  <body>
9      <div class="outer">
10
11     </div>
12
13     <script>
14         const outerDOM = document.querySelector('.outer')
15         const imgElement = document.createElement('img');
16
17         // Create Element
18         imgElement.src = '../img/cf-logo.png';
19         imgElement.alt = 'Coding Factory';
20
21         // Append the new element to DOM as a child of 'outer'
22         outerDOM.append(imgElement)
23     </script>
24 </body>
25 </html>
```

HTML Attributes (2)

- Το Element object ορίζει επίσης τις μεθόδους **getAttribute()** , **setAttribute()**, **hasAttribute()** και **removeAttribute()** για να διαχειριζόμαστε τα attributes των HTML Elements και να μπορούμε να πάρουμε την τιμή του attribute, να κάνουμε set την τιμή ενός attribute (αν δεν υπάρχει, δημιουργείται), να διαγράφουμε attributes, να ελέγξουμε αν υπάρχει ένα attribute, αντίστοιχα

Attribute / Properties naming

- Οι κανόνες αντιστοίχισης των ονομάτων των HTML attributes σε DOM properties, γίνεται γενικά διατηρώντας τα HTML attributes στα οποία το convention είναι με πεζά γράμματα (lowercase) Για παράδειγμα style, value, src.
- Τα properties του DOM είναι case-sensitive και camelCase
- Εξαίρεση αποτελούν τα event-handler properties, όπως onclick που γράφονται με lowercase

Properties values

- Τα DOM properties που αντιστοιχούν στα HTML attributes γενικά έχουν ως τιμές strings
- Εκτός αν το html attribute είναι boolean (π.χ. checked) ή number (π.χ. maxLength) ή συνάρτηση (π.χ. onclick)

Attribute names και JS reserved words

- Κάποια html attributes έχουν conflicts με JS reserved words. Τότε το αντίστοιχο DOM object property γίνεται prefix με το πρόθεμα ‘html’
- Για παράδειγμα η attribute for του <label> γίνεται *htmlFor*
- Το attribute *class* αποτελεί εξαίρεση και γίνεται **className**

Class attribute (1)

- Το class attribute είναι σημαντικό γιατί είναι ο βασικός τρόπος να προσδώσουμε styling στα HTML αντικείμενα
- Το property className είναι string, και δεν εξυπηρετεί την δυναμική εισαγωγή και διαγραφή class-names σε ένα element
- Για αυτό το Element object μας παρέχει την ιδιότητα **classList** που είναι DOMTokenList (δυναμική λίστα σαν πίνακας) που με τη σειρά της μας παρέχει τις μεθόδους **add(), remove(), contains() και toggle()**

Class attribute (2)

- Αν για παράδειγμα έχουμε μία κλάση στο CSS μας με όνομα .hidden { display: none } και θέλουμε να εμφανίζουμε και να αποκρύπτουμε ένα στοιχείο, έστω <p>, έστω myP, τότε για να αποκρύψουμε:
- *myP.classList.add(ʻhidden’)*
- Ќαι να εμφανίσουμε με:
- *myP.classList.remove(‘hidden’)*

Class attribute (3)

- myP.classList.toggle('hidden')
- Η toggle() πρώτα προσθέτει και μετά αφαιρεί εναλλάξ την κλάση που λαμβάνει ως παράμετρο
- Αν ένα element έχει μία αρχική κλάση και κάνουμε toggle μία άλλη κλάση με τις ίδιες CSS ιδιότητες θα εφαρμοστεί εκείνη η κλάση με το υψηλότερο specificity

Attribute Default values

- Για κάποια στοιχεία οι HTML ιδιότητες αντιστοιχούνται λίγο διαφορετικά. Για παράδειγμα στα input στοιχεία η *value* attribute ενός text input αντιστοιχεί στην ***defaultValue*** στο DOM ενώ το ***value*** property είναι η *current value* του input στοιχείου
- Το ίδιο ισχύει και την checked, defaultChecked

data-* attributes

- Μπορούμε σε κάθε HTML element να ορίσουμε custom dataset attributes που το όνομά τους ξεκινά από data-*
- Στο DOM τα Elements έχουν ένα **dataset** property που είναι object, με properties που αντιστοιχούν στο data-* property με το prefix data- , δηλαδή οτιδήποτε έχουμε δηλώσει μετά (αν για παράδειγμα δώσουμε data-hello-world το αντίστοιχο property είναι helloWorld)
- Αν θέλαμε για παράδειγμα να εμφανίζουμε βαθμολογίες σε κλίμακα / 100
- Βαθμολογία
- .rating::after { content: " / " attr(data-scale); }
- let ratingScale = document.querySelector("rating").dataset.scale

Εισαγωγή στοιχείων HTML

- **document.createElement(tagName)** – Δημιουργεί ένα HTML Element τύπου tagName.

  Π.χ. const newDiv = document.createElement("div");

- **element.remove()** ή **parent.removeChild(child)** – Διαγράφει ένα element

- **.append** και **.appendChild(aChild)** – Προσθέτει ως child node ένα Element σε ένα parent element, στο τέλος της λίστας των children του Parent Node. Π.χ. const paragraph = document.body.appendChild(document.createElement('p'));

- **.setAttribute(name, value)** – Θέτει (updates) την τιμή ενός attribute του Element. Αν το attribute δεν υπάρχει, ένα νέο attribute δημιουργείται με την συγκεκριμένη τιμή. Π.χ. button.setAttribute("name", "helloButton");

Προσθήκη children (1)

- **append(node1, node2, string, ..)** προσθέτει ένα ή περισσότερα nodes ή/και strings (τα strings μετατρέπονται σεText nodes) στο τέλος του Element ως last children.
- Π.χ. let div = document.createElement("div"); let p = document.createElement("p"); div.append(p)
- const div = document.createElement("div") div.append("Some text")
- **.prepend (node1, node2, string, ..)** το ίδιο με την append, προσθέτει στη αρχή ως first children

Προσθήκη children (2)

- **after (node1, nod2, string, ..)** όπως πριν, προσθέτει αμέσως μετά ένα sibling node
- **.before (node1, nod2, string, ..)** όπως παραπάνω αμέσως πριν ένα sibling node

Περιεχόμενο

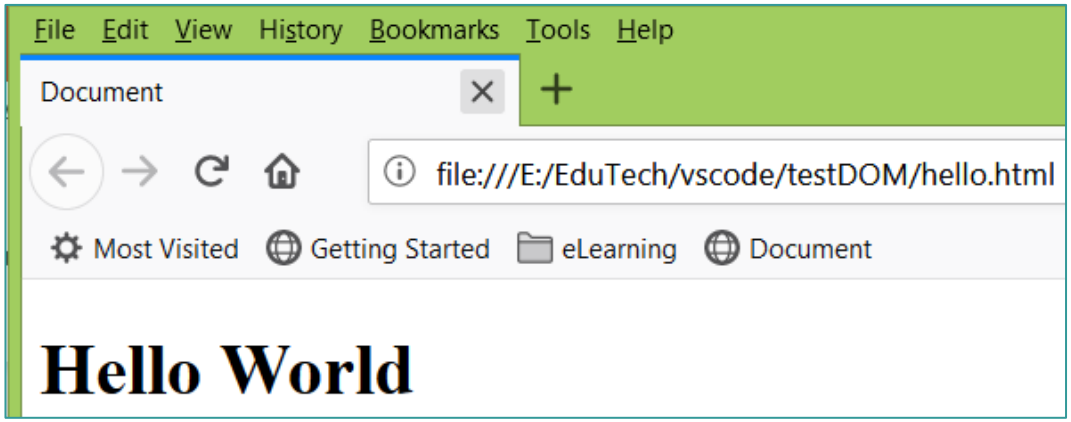
- Για τη διαχείριση του περιεχομένου των στοιχείων HTML υπάρχουν οι παρακάτω ιδιότητες:
  - **.innerHTML** – που λαμβάνει υπόψη και τυχόν tags μορφοποιήσεων (π.χ. <b>) ή και στοιχείων όπως <p> κλπ. που περιλαμβάνονται στο περιεχόμενο. Είναι η πιο συχνά χρησιμοποιούμενη ιδιότητα για διαχείριση κειμένου
  - **.outerHTML** – Περιλαμβάνει το tag και του ίδιου του αντικειμένου καθώς και τους απογόνους
  - **textContent** – διερμηνεύει ως απλό κείμενο. Σε Element nodes επιστρέφει το κείμενο όλων των απογόνων του Element

HTML DOM - innerHTML

hello.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <h1 id="helloText"></h1>
10
11      <script>
12          document.getElementById('helloText').innerHTML = "Hello World";
13      </script>
14  </body>
15  </html>
```

- Όπως είπαμε βάζουμε το script **μετά το <h1>** στο τέλος του <body> ώστε να έχει φορτωθεί πρώτα στο DOM η σελίδα
- Το αποτέλεσμα είναι αυτό που φαίνεται στον browser



DOM – Dynamic content (1)


```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" cont
6      <title>Document</title>
7  </head>
8  <body>
9      <h1 id="title"></h1>
10     <p class="aueb-text"></p>
11     <div class="main"></div>
12
13     <script src="./dom1.js">
14     </script>
15 </body>
16 </html>
```

- Η HTML σελίδα δεν έχει περιεχόμενο
- Θα δώσουμε δυναμικά μέσω του DOM

DOM – Dynamic content (2)


```
1  const titleDOM = document.getElementById('title')
2  const auebTextDOM = document.querySelector('.aueb-text')
3  const main = document.querySelector('.main')
4  let image = 'https://codingfactory.aueb.gr/sites/all/themes/cf_theme/logo.png'
5  let cfName = 'Coding Factory'
```

- Επιλέγουμε τα στοιχεία του HTML και αντιστοιχούμε σε DOM elements
- Ορίζουμε και δύο μεταβλητές image, και name

DOM – Dynamic content (3)


```
titleDOM.textContent = "Coding Factory"
auebTextDOM.innerHTML = `Το Coding Factory είναι ένα εκπαιδευτικό
πρόγραμμα στην Ανάπτυξη Λογισμικού του Κέντρου Δια Βίου Μάθησης του
Οικονομικού Πανεπιστημίου Αθηνών (ΟΠΑ) και στη συνέχεια διασύνδεσης
των αποφοίτων με συνεργαζόμενες επιχειρήσεις σε όλη την Ελλάδα.
Η συνολική διάρκεια του προγράμματος είναι έξι (6) μήνες. `
main.innerHTML = `<article class="course">
  <div class="cf-container">
    <img src="${image}" class="cf-img img" alt="${cfName}" />
  </div>
  <footer>
    <p class="course-name">&copy; AUEB - Coding Factory</p>
  </footer>
</article>`
```

- Παρατηρούμε ότι η δομή είναι πιο πολύπλοκη είναι πιο εύκολο να δημιουργούμε nodes με HTML μέσα στο innerHTML παρά με createElement

Είσοδος / Έξοδος με .value

- Για το διάβασμα και την εκχώρηση κειμένου στην ιδιότητα value, HTML στοιχείων σε forms όπως textbox χρησιμοποιούμε την ιδιότητα .defaultValue και .value

Στατικό Template


```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-
6       <meta name="viewport" c
7       <title>Document</title>
8   </head>
9   <body>
10      <p>Hello World!</p>
11  </body>
12  </html>
```

- Έχουμε δύο στοιχεία
  - Ένα <p>
  - Ένα child του <p> που είναι Text node με τιμή Hello World!

Δυναμικό περιεχόμενο text


```
 9    <body>
10        <p></p>
11
12        <script src="./js/helloworld.js"></script>
13    </body>
```

helloworld.js

testbed > chapter14 > js > JS helloworld.js > ...

```
1    const p = document.querySelector('p')
2    p.textContent = "Hello World"
3
```

- querySelector και textContent. Η ιδιότητα textContent δεν κάνει parse, HTML

innerHTML

testbed > chapter14 > helloworld.html > html > body

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge"
    <meta name="viewport" content="width=device-width,
    <title>Hello</title>
    <link rel="stylesheet" href="./css/helloworld.css">
</head>
```

```
10      <div class="container">
11
12      </div>
13
14
15      <script src="./js/helloworld.js"></sc
16  </body>
17  </html>
```

testbed > chapter14 > css > # helloworld.css

```
1   .container {
2       width: 90%;
3       border: 1px solid black;
4       margin-left: auto;
5       margin-right: auto;
6   }
```

JS helloworld.js

testbed > chapter14 > js > JS helloworld.js > ...

```
1   const container = document.querySelector('.container')
2   container.innerHTML = "<p>Hello World !!!</p>"
```

- Η ιδιότητα .innerHTML κάνει parse και HTML

+= innerHtml


```
 9   <body>
10       <div class="container">
11           <p><strong>Coding Factory</strong></p>
12       </div>
13
14       <script src="./js/helloworld.js"></script>
15   </body>
16   </html>
17
```

```
1   .container {
2       width: 90%;
3       border: 1px solid black;
4       margin-left: auto;
5       margin-right: auto;
6   }
```

helloworld.js

testbed > chapter14 > js > JS helloworld.js > ...

```
1   const container = document.querySelector('.container')
2   container.innerHTML += "<p>Says Hello World !!!</p>"
```

- Μπορούμε να διαβάσουμε το innerHTML και να προσθέσουμε στη στο τέλος

appendChild


```
<body>
    <div class="container">

    </div>

    <script src="./js/helloworld.js"></script>
</body>
</html>
```

```
.container {
    width: 90%;
    border: 1px solid black;
    margin-left: auto;
    margin-right: auto;
}
```

helloworld.js

estbed > chapter14 > js > JS helloworld.js > ...

```
const container = document.querySelector('.container')
const p = document.createElement('p')
p.textContent = "Hello World !"

container.appendChild(p)
```

- Το ίδιο όπως πριν. Με appendChild δεν είναι τόσο readable όσο πριν

Append vs AppendChild


```html
<body>
    <div class="container">

    </div>

    <script src="./js/helloworld.js"></script>
</body>
</html>
```

```css
.container {
    width: 90%;
    border: 1px solid black;
    margin-left: auto;
    margin-right: auto;
}
```

helloworld.js

testbed > chapter14 > js > helloworld.js > ...

```js
const container = document.querySelector('.container')
const p = document.createElement('p')
const text = "Hello Coding Factory!"

p.append(text)
container.appendChild(p)
```

- Η append προσθέτει ένα ή περισσότερα node objects ή/και text nodes εν αντιθέσει με την appendChild που προσθέτει μόνο ένα Element node

Styling (1)


```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, in
7       <title>Document</title>
8   </head>
9   <body>
10
11      <script>
12          document.body.style.backgroundColor = '#ff0000'
13      </script>
14  </body>
15  </html>
```

- Με **.style.<cssPoperty>** δίνουμε inline styling
- Αν η CSS ιδιότητα περιέχει δύο λέξεις όπως background-color τότε η JS property είναι camelCase, δηλαδή backgroundColor

Styling (2)


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=e
    <meta name="viewport" content="width=device-widt
    <title>Document</title>
</head>
<body>
    <p class="hello">Hello</p>

    <script>
        function formatAndPosition(e, x, y) {
            e.style.width = '100px'
            e.style.textAlign = 'center'
            e.style.border = '1px solid black'
            e.style.position = 'absolute'
            e.style.left = `${x}px`
            e.style.top = `${y}px`
        }

        let par = document.querySelector('.hello')
        formatAndPosition(par, 150, 100)
    </script>
</body>
</html>
```

- Αν και δεν συνηθίζουμε να δίνουμε styling με την ιδιότητα style, κάποιες φορές μπορεί να χρειάζεται
- Όπως όταν δεν ξέρουμε από πριν την θέση για absolute positioning ενός αντικειμένου

classList (1)


```
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge"
6       <meta name="viewport" content="width=device-width, i
7       <title>Hello</title>
8       <link rel="stylesheet" href="./css/helloworld.css">
9   </head>
10  <body>
11      <div id="container">
12
13      </div>
14
15      <script src="./js/helloworld.js"></script>
16  </body>
```

```
1   .container {
2       width: 90%;
3       border: 1px solid black;
4       margin-left: auto;
5       margin-right: auto;
6       /* background-color: red; */
7   }
8
9   /* p {
10      text-align: center;
11  } */
```

JS helloworld.js

testbed > chapter14 > js > JS helloworld.js > ...

```
2   container.classList.add('container')
3   container.style.backgroundColor = 'red'
4
5   const p = document.createElement('p')
6   p.append("Hello!")
7   p.style.textAlign = 'center'
8
9   container.append(p)
```

- Με την ιδιότητα style (γρ. 3, 7) που είναι inline property μπορούμε να δώσουμε styling. Καλύτερα όμως με classList.add και css file που είναι γενικός μηχανισμός

classList (2)

testbed > chapter14 > helloworld.html > html > body > div#container

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge"
6       <meta name="viewport" content="width=device-width, i
7       <title>Hello</title>
8       <link rel="stylesheet" href="./css/helloworld.css">
9   </head>
10  <body>
11      <div id="container">
12
13      </div>
```

testbed > chapter14 > css > # helloworld.css >

```
1   .container {
2       width: 90%;
3       border: 1px solid black;
4       margin-left: auto;
5       margin-right: auto;
6   }
```

JS helloworld.js

testbed > chapter14 > js > JS helloworld.js > ...

```
1   const container = document.querySelector('#container')
2   container.classList.add('container')
3   container.innerHTML = '<p>Hello Coding!</p>'
```

- Με classList.add() προσθέτουμε δυναμικά κλάσεις από το css

classList (3)

testbed > chapter14 > helloworld.html > html > body > div#container

```
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge"
6       <meta name="viewport" content="width=device-width, i
7       <title>Hello</title>
8       <link rel="stylesheet" href="./css/helloworld.css">
9   </head>
10  <body>
11      <div id="container">
12
13      </div>
14
15      <script src="./js/helloworld.js"></script>
16  </body>
```

testbed > chapter14 > css > # helloworld.css > .container

```
1   .container {
2       width: 90%;
3       border: 1px solid black;
4       margin-left: auto;
5       margin-right: auto;
6       background-color: rgb(84, 200, 91);
7   }
8
9   .text-center {
10      text-align: center;
11  }
```

JS helloworld.js

testbed > chapter14 > js > JS helloworld.js > p

```
1   const container = document.querySelector('#container')
2   container.classList.add('container')
3
4   const p = document.createElement('p')
5   p.append("Hello!")
6   p.classList.add('text-center')
7
8   container.append(p)
```

- Με classList.add() εισάγουμε δυναμικά την κλάση container, text-center

Multiple classList

testbed > chapter14 > helloworld.html > html > body > div#container

```
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge"
6       <meta name="viewport" content="width=device-width, i
7       <title>Hello</title>
8       <link rel="stylesheet" href="./css/helloworld.css">
9   </head>
10  <body>
11      <div id="container">
12
13      </div>
14
15      <script src="./js/helloworld.js"></script>
16  </body>
17  </html>
```

testbed > chapter14 > css > # helloworld.css > .text-center

```
1   .container {
2       width: 90%;
3       border: 1px solid black;
4       background-color: rgb(84, 200, 91);
5   }
6
7   .box-center {
8       margin-left: auto;
9       margin-right: auto;
10  }
11
12  .text-center {
13      text-align: center;
14  }
```

JS helloworld.js

testbed > chapter14 > js > JS helloworld.js > ...

```
1   const container = document.querySelector('#container')
2   container.classList.add('container', 'box-center')
3   const p = document.createElement('p')
4   p.append("Hello!")
5   p.classList.add('text-center')
6
7   container.append(p)
```

- Με classList προσθέτουμε περισσότερες από μία κλάσεις

Array destucturing

testbed > chapter14 > helloworld.html > html > body > div#container

```
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge"
6       <meta name="viewport" content="width=device-width, i
7       <title>Hello</title>
8       <link rel="stylesheet" href="./css/helloworld.css">
9   </head>
10  <body>
11      <div id="container">
12
13      </div>
14
15      <script src="./js/helloworld.js"></script>
16  </body>
17  </html>
```

testbed > chapter14 > css > # helloworld.css > .box-center

```
1   .container {
2       width: 90%;
3       border: 1px solid black;
4       background-color: rgb(84, 200, 91);
5   }
6
7   .box-center {
8       margin-left: auto;
9       margin-right: auto;
10  }
11
12  .text-center {
13      text-align: center;
14  }
```

helloworld.js

testbed > chapter14 > js > JS helloworld.js > ...

```
1   const container = document.querySelector('#container')
2   const cls = ['container', 'box-center']
3   container.classList.add(...cls)
4   const p = document.createElement('p')
5   p.append("Hello!")
6   p.classList.add('text-center')
```

- Ο ίδιος μηχανισμός της classList με array destructuring

Onclick event


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-sca
    <title>Document</title>
</head>
<body>
    <button type="button" onclick="sayHello()">Say Hello</button>

    <script src="./js/btnhello.js"></script>
</body>
```

btnhello.js

testbed > chapter14 > js > JS btnhello.js > ...

```js
function sayHello() {
    alert('Hello Coding Factory')
}
```

- Η onclick είναι μία ιδιότητα του Element στοDOM
- Ενεργοποιείται όταν γίνει κλικ στο element όπως εδώ το button

Παράδειγμα


```
<body>

    <script>
        let btn = document.createElement('button')
        btn.setAttribute('type', 'button')
        btn.innerHTML = 'Click Me'
        btn.setAttribute('onclick', 'console.log("Hello")')

        let div = document.createElement('div')
        div.appendChild(btn)
        document.body.append(div)
    </script>
</body>
</html>
```

- Δημιουργεί ένα button και ένα div με δυναμικό τρόπο. Προσθέτει μία ιδιότητα onclick στο button. Στο onclick εκτυπώνει hello στην κονσόλα

Clones


```
 9   <body>
10       <p class="hello-text">
11           Hello CF!
12       </p>
13
14       <script src="./js/clone.js"></script>
15   </body>
```

clone.js

testbed > chapter14 > js > JS clone.js > ...

```
1   const helloText = document.querySelector('.hello-text')
2   const helloCloned = helloText.cloneNode(true)
3   document.body.appendChild(helloCloned)
```

- Αν εισάγουμε (appendChild) ένα ήδη υπάρχον στοιχείο του DOM σε άλλη θέση, μετακινείται. Αν θέλουμε, μπορούμε να δημιουργήσουμε ένα **clone του Element** και να το εισάγουμε στο DOM χωρίς να επηρεάσουμε τον αρχικό κόμβο
- Η παράμετρος της cloneNode(deep) ορίζει αν θα αντιγραφεί όλο το nested μέρος του node (true) ή μόνο το node (false)

Events και Γεγονοστρεφής Προγραμματισμός

- Όπως έχουμε αναφέρει **γεγονός (event)** είναι μία αλλαγή στην κατάσταση ενός αντικειμένου που μπορεί να προέρχεται είτε από κάποια ενέργεια του χρήστη (π.χ. κλικ σε button) ή και από το σύστημα (π.χ. το φόρτωμα της web σελίδας)
- Όταν συμβεί ένα event μπορούμε να το χειριστούμε (handle) με κάποια συνάρτηση που ονομάζεται Event Handler

HTML onclick attribute


```
 9   <body>
10       <button type="button" onclick="sayHello()">Say Hello</button>
11
12       <script src="./js/btnhello.js"></script>
13   </body>
14   </html>
```

btnhello.js

testbed > chapter14 > js > JS btnhello.js > ...

```
1   function sayHello() {
2       alert('Hello Coding Factory')
3   }
```

- Όπως έχουμε δει μπορούμε να χρησιμοποιήσουμε την ιδιότητα onclick. Ωστόσο δεν είναι ο προτεινόμενος τρόπος γιατί δεν μπορούμε να κάνουμε register περισσότερα από ένα events στο element

Event Handlers

- Υπάρχουν δύο τρόποι να ορίσουμε event Handlers με την JavaScript

Μέσω των events όπως κάναμε με την ιδιότητα onclick, δηλαδή ορίζουμε έναν event handler ως εξής:

```
<script>
    let helloBtn = document.getElementById('hello-btn');
    helloBtn.onclick = function() {
        alert('Hello');
    }
</script>
```

Μέσω ενός Event Listener, όπως βλέπουμε δεξιά
Οι Event Listeners είναι πιο γενικός τρόπος ορισμού event handlers

```
<script>
    let helloBtn = document.getElementById('hello-btn');
    helloBtn.addEventListener('click', function() {
        alert('Hello World!');
    });
</script>
```

Event Handlers - Ιδιότητα onclick


```
<script>
    let helloBtn = document.getElementById('hello-btn');
    helloBtn.onclick = function() {
        alert('Hello');
    }
</script>
```

- Ο Event Handler ορίζεται με JavaScript στην ιδιότητα *onclick* (καθώς και μία ανώνυμη function)
- Πρόκειται για παρόμοιο μηχανισμό με τον ορισμό της ιδιότητας *onclick* στην ιδιότητα ενός στοιχείου HTML, όπως το button

Event Handlers – Event Listeners (1)


```
<script>
    let helloBtn = document.getElementById('hello-btn');
    helloBtn.addEventListener('click', function() {
        alert('Hello World!');
    });
</script>
```

- Οι Event Listeners είναι ένα γενικός μηχανισμός που δίνει τη δυνατότητα να ορίζουμε όχι μόνο ένα αλλά πολλούς handlers σε διάφορα events ενός στοιχείου HTML ενώ δίνει και τη δυνατότητα να ορίζουμε και την προτεραιότητα εκτέλεσης των event handlers σε περίπτωση που έχουμε nested elements

Event Handlers – Event Listeners (2)


```
<script>
    let helloBtn = document.getElementById('hello-btn');
    helloBtn.addEventListener('click', function() {
        alert('Hello World!');
    });
</script>
```

- Η addEventListener(eventType, eventHandler, [useCapture]) λαμβάνει δύο υποχρεωτικές παραμέτρους (eventType και EventHandler) και μία προαιρετική (useCapture)
- Το eventType είναι ο τύπος του event, ο event handler είναι μία callback συνάρτηση που χειρίζεται το event και το useCapture είναι μία boolean μεταβλητή που by default είναι false και ορίζει την προτεραιότητα εκτέλεσης των event handlers στην περίπτωση που έχουμε nested elements

Bubbling & Capturing

- Ας υποθέσουμε ότι έχουμε ένα με ένα event handler onclick και ένα <button> μέσα στο με ένα άλλο event handler πάλι onclick
- Αν κάνουμε κλικ πάνω στο button (που είναι όμως και κλικ πάνω στο) ποιος event handler θα εκτελεστεί πρώτος?
- **Αν το useCapture είναι true**, τότε πρώτα εκτελείται ο event handler του και μετά ο event handler του button (Capturing)
- Αν το useCapture είναι false, τότε πρώτα εκτελείται ο event handler του button και μετά ο event handler του (Bubbling)
- Το **default είναι false**, και χρησιμοποιείται στις περισσότερες περιπτώσεις

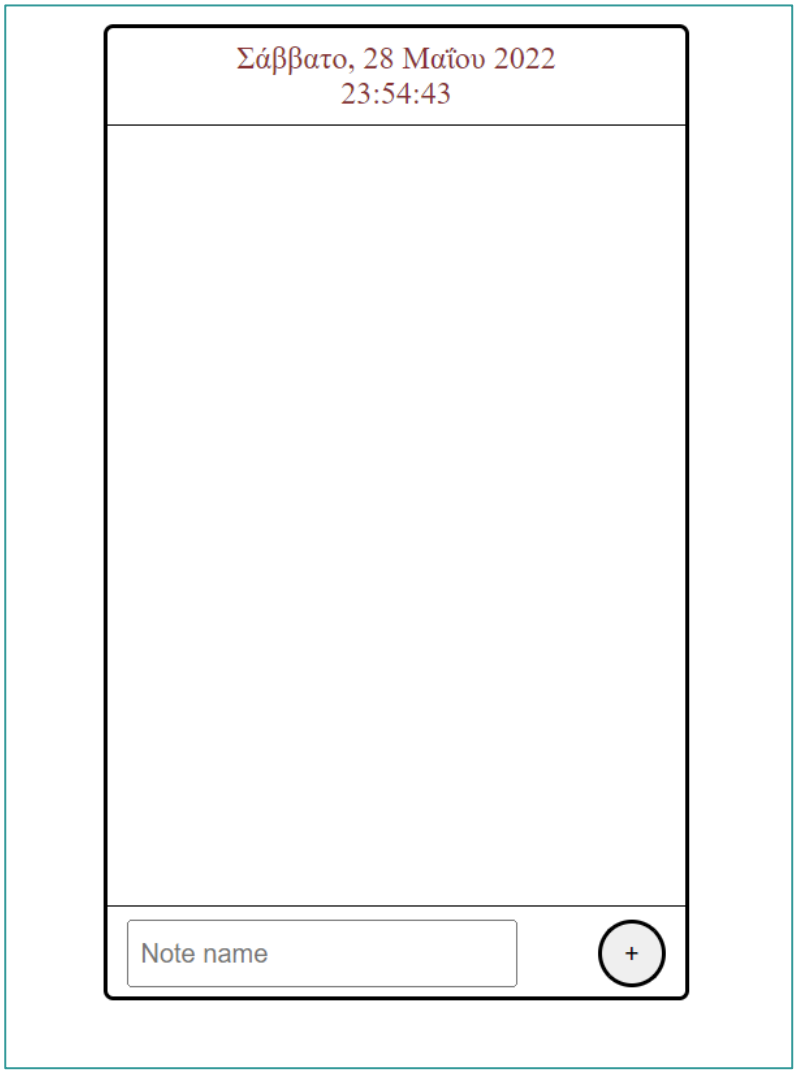
Τύποι Events

- Υπάρχουν πολλοί τύποι events: https://developer.mozilla.org/en-US/docs/Web/Events
- *Μερικά χρήσιμα events είναι τα παρακάτω:*
  - *click – έχει γίνει κλικ στο HTML Element*
  - *input – όταν το value ενός <input>, <select>, <textarea> αλλάξει*
  - *focus – Ένα στοιχείο HTML έχει γίνει focus (για παράδειγμα έχουν κάνει κλικ μέσα σε ένα text box)*
  - *blur - Ένα στοιχείο HTML έχει χάσει το focus*
  - *change – Ενεργοποιείται όταν ένα <input>, <select> ή <textarea> αλλάξει τιμή (και το lose focus είναι change)*
  - *keydown – Οποιοδήποτε κουμπί του πληκτρολογίου έχει πατηθεί*
  - *keyup - Οποιοδήποτε κουμπί του πληκτρολογίου έχει πατηθεί και αφεθεί (released)*
  - *load – Ένας πόρος, όπως π.χ. η σελίδα, έχει φορτωθεί*

Event Listeners / Event handlers

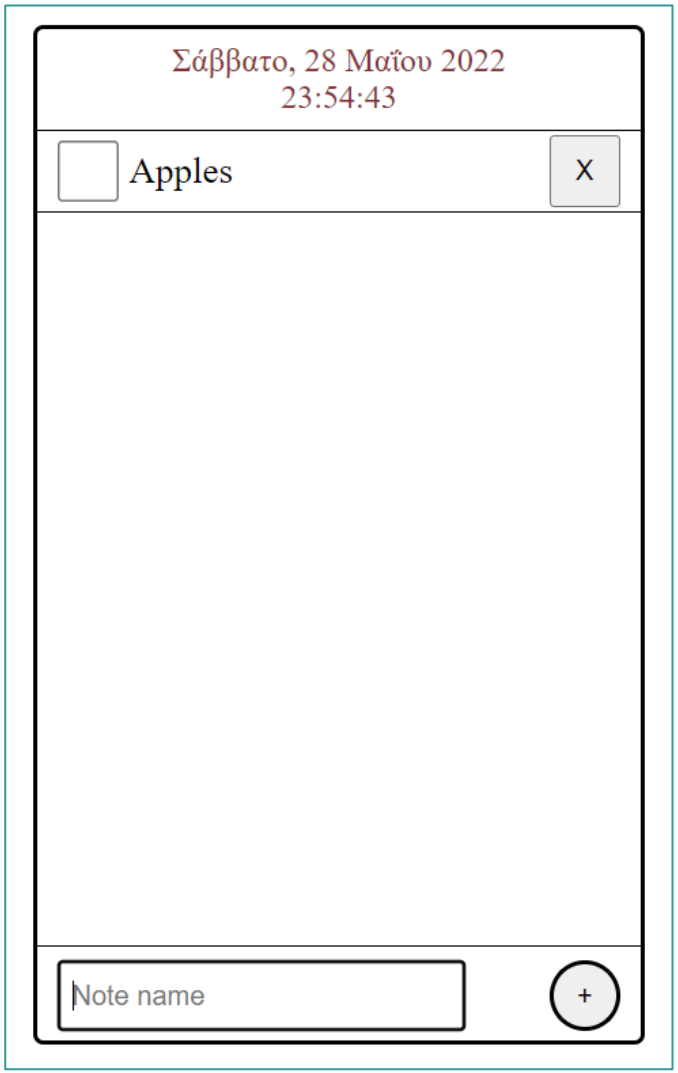
- Οι **Event Handlers** ή **Event Listeners** μπορεί να είναι μία κανονική συνάρτηση ή μία ανώνυμη συνάρτηση
- Η χρήση **ανώνυμων συναρτήσεων** ως event handlers είναι **κλασσική περίπτωση χρήσης ανώνυμων συναρτήσεων** δεδομένου ότι συνήθως οι handlers χρησιμοποιούνται μόνο εκεί, μόνο μέσα δηλαδή στον Event Listener, επομένως δεν έχει νόημα να οριστούν ως named functions, αφού δεν χρησιμοποιούνται κάπου αλλού

Εργασία - Εφαρμογή Σημειώσεων (1)



- Έστω ότι θέλουμε να φτιάξουμε μία εφαρμογή κράτησης σημειώσεων, όπου ο χρήστης θα μπορεί να προσθέτει και να διαγράφει σημειώσεις αλλά και να σημειώνει ως περατωμένες με διακριτή διαγραφή

Εργασία – Εφαρμογή Σημειώσεων (2)



- Ημερομηνία
- Χώρος σημειώσεων
  - Κάθε row αποτελείται από ένα checkbox, ένα label και ένα button
- Χώρος εισαγωγής σημείωσης
  - Αποτελείται από ένα textbox και ένα button