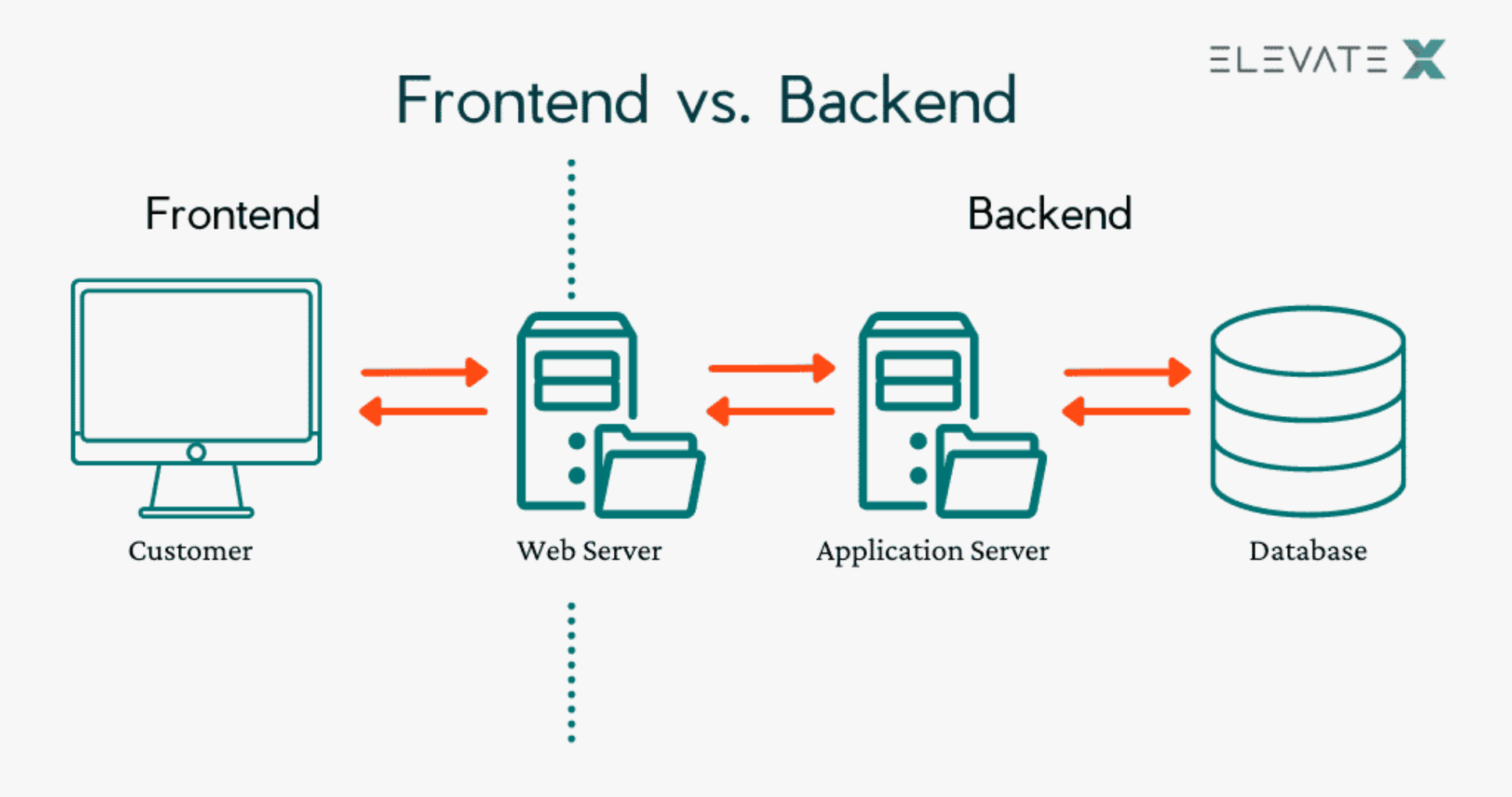
Σημασιολογικός και Κοινωνικός Ιστός

Διάλεξη 03 – Δεδομένα και σήμανση: HTML, Markdown, CSV, JSON, XML

Γεώργιος Δημητρακόπουλος
dimitrakopoulos@ionio.gr

Τμήμα Πληροφορικής
Ιόνιο Πανεπιστήμιο

Front-end vs back-end



Front-End

- HTML
- https://www.w3schools.com/html/

- CSS for style

- Javascript for dynamic content

HTML

- **H**yper**T**ext **M**arkup **L**anguage

- Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «<» και «>» (π.χ. <html>).

- Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (π.χ. <h1> και </h1>). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

- Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και να τα συνθέσει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να παρουσιάσει το περιεχόμενο της σελίδας.

HTML - Ιστορικό

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1
2017	W3C Recommendation: HTML5.1 2nd Edition
2017	W3C Recommendation: HTML5.2

HTML 5 - minimum document

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

Περιεχόμενο

</body>
</html>
```

Tags

- Headings <h1>-<h6>
- Style: <b>,<i>,<u>,<sub>,<sup>,..
- Paragraph <p>, break
- Links <a>, images <img>
- Tables <tr>,<tr>,<th>,<td>
- Lists <ul>,<ol>,
- Forms <form>, <input>, <select>, <option>
- Other: , ,…

attributes

- Τμ. Πληροφορικής
- Tag: a
- Attribute: href
- Attribute value: "di.ionio.gr"

Markdown

- Σύντομος τρόπος σήμανσης μορφοποίησης κειμένου
- Διαβάζεται και ως απλό κείμενο
- Χρήσιμο για readme κώδικα, π.χ. Github.
- https://www.markdownguide.org/basic-syntax/

Markdown

- Headings

Markdown	HTML	Rendered Output
`# Heading level 1`	`<h1>Heading level 1</h1>`	Heading level 1
`## Heading level 2`	`<h2>Heading level 2</h2>`	Heading level 2
`### Heading level 3`	`<h3>Heading level 3</h3>`	Heading level 3
`#### Heading level 4`	`<h4>Heading level 4</h4>`	Heading level 4
`##### Heading level 5`	`<h5>Heading level 5</h5>`	Heading level 5
`###### Heading level 6`	`<h6>Heading level 6</h6>`	Heading level 6

- Εναλλακτική Σύνταξη

Markdown	HTML	Rendered Output
`Heading level 1` `==========`	`<h1>Heading level 1</h1>`	Heading level 1
`Heading level 2` `----------`	`<h2>Heading level 2</h2>`	Heading level 2

Markdown

Paragraphs

To create paragraphs, use a blank line to separate one or more lines of text.

Markdown	HTML	Rendered Output

`I really like using Markdown.`
 	
`I think I'll use it to format`
`all of my documents from now on.`
	`<p>I really like using Markdown.`
	`</p>`
	
	`<p>I think I'll use it to format`
	`all of my documents from now on.`
	`</p>`
		I really like using Markdown.
		
		I think I'll use it to format all of my documents from now on.

Line Breaks

To create a line break or new line (`<br>`), end a line with two or more spaces, and then type return.

Markdown	HTML	Rendered Output
`This is the first line.` `And this is the second line.`	`<p>This is the first line. ` `And this is the second line.</p>`	This is the first line. And this is the second line.

Bold

To bold text, add two asterisks or underscores before and after a word or phrase. To bold the middle of a word for emphasis, add two asterisks without spaces around the letters.

Markdown	HTML	Rendered Output
`I just love **bold text**.`	`I just love <strong>bold text</strong>.`	I just love **bold text**.
`I just love __bold text__.`	`I just love <strong>bold text</strong>.`	I just love **bold text**.

Italic

To italicize text, add one asterisk or underscore before and after a word or phrase. To italicize the middle of a word for emphasis, add one asterisk without spaces around the letters.

Markdown	HTML	Rendered Output
`Italicized text is the *cat's meow*.`	`Italicized text is the <em>cat's meow</em>.`	Italicized text is the *cat's meow*.
`Italicized text is the _cat's meow_.`	`Italicized text is the <em>cat's meow</em>.`	Italicized text is the *cat's meow*.

Blockquotes

To create a blockquote, add a > in front of a paragraph.

```
> Dorothy followed her through many of the beautiful rooms in her castle.
```

The rendered output looks like this:

> Dorothy followed her through many of the beautiful rooms in her castle.

Markdown

Ordered Lists

To create an ordered list, add line items with numbers followed by periods. The numbers don't have to be in numerical order, but the list should start with the number one.

Markdown	HTML	Rendered Output

`1. First item`
`2. Second item`
`3. Third item`
`4. Fourth item`
	`<ol>`
	`  <li>First item</li>`
	`  <li>Second item</li>`
	`  <li>Third item</li>`
	`  <li>Fourth item</li>`
	`</ol>`
		1. First item
		2. Second item
		3. Third item
		4. Fourth item

Unordered Lists

To create an unordered list, add dashes (-), asterisks (*), or plus signs (+) in front of line items. Indent one or more items to create a nested list.

Markdown	HTML	Rendered Output

`- First item`
`- Second item`
`- Third item`
`- Fourth item`
	`<ul>`
	`  <li>First item</li>`
	`  <li>Second item</li>`
	`  <li>Third item</li>`
	`  <li>Fourth item</li>`
	`</ul>`
		• First item
		• Second item
		• Third item
		• Fourth item

Code

To denote a word or phrase as code, enclose it in backticks (`).

Markdown	HTML	Rendered Output
`At the command prompt, type `nano`.`	`At the command prompt, type <code>nano</code>.`	At the command prompt, type `nano`.

Links

To create a link, enclose the link text in brackets (e.g., `[Duck Duck Go]`) and then follow it immediately with the URL in parentheses (e.g., `(https://duckduckgo.com)`).

```
My favorite search engine is [Duck Duck Go](https://duckduckgo.com).
```

Images

To add an image, add an exclamation mark (!), followed by alt text in brackets, and the path or URL to the image asset in parentheses. You can optionally add a title in quotation marks after the path or URL.

```
![The San Juan Mountains are beautiful!](/assets/images/san-juan-mountains.jpg "San Juan Mountains")
```

Περιγραφή Δεδομένων - CSV

- Comma Separated Values
- Κείμενο σε μορφή πίνακα
- Στήλες χωρίζονται με κάποιο χαρακτήρα, π.χ. Tab, Κόμμα (στο ελληνικό excel, επειδή το , χρησιμοποιείται ως υποδιαστολή για τα δεκαδικά ψηφία, χρησιμοποιείται το ;)
- Κάθε γραμμή του πίνακα τελειώνει με \n
- Μπορεί η πρώτη γραμμή να είναι τα ονόματα των στηλών
- Για κείμενο, συνηθίζεται να περικλείεται σε εισαγωγικά (μπορεί να περιέχει κόμμα)

Περιγραφή Δεδομένων - JSON

- JavaScript Object Notation
- Μορφή κειμένου
- Δομημένα δεδομένα, π.χ. αντικείμενα
- Σύνταξη:
  - Αντικείμενο: {}
  - Ιδιότητα: τιμή
  - Λίστα: []

Παράδειγμα JSON

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "powers": [
                    "Radiation resistance",
                    "Turning tiny",
                     "Radiation blast«
                    ]
    },
    {
      "name": "Madame Uppercut",
      "age": 39,
      "powers": [       "Million tonne punch",       "Damage resistance",       "Superhuman reflexes"]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
     "powers": [     "Immortality",       "Heat Immunity",       "Inferno",       "Teleportation",       "Interdimensional travel"       ]
    }
  ]
}
```

Χρήση αντικειμένο JSON

- superHeroes.homeTown
- superHeroes['active’]
- superHeroes['members'][1]['powers'][2]

XML

- **Extensible Markup Language**
- Η XML είναι μια γλώσσα σήμανσης και μια μορφή αρχείου για την αποθήκευση, τη μετάδοση και την ανακατασκευή αυθαίρετων δεδομένων. Ορίζει ένα σύνολο κανόνων για την κωδικοποίηση εγγράφων σε μορφή που είναι αναγνώσιμη τόσο από τον άνθρωπο όσο και από τις μηχανές. Η προδιαγραφή XML 1.0 του World Wide Web Consortium του 1998 και πολλές άλλες σχετικές προδιαγραφές—όλες δωρεάν ανοιχτά πρότυπα—καθορίζουν την XML.

- Οι σχεδιαστικοί στόχοι της XML δίνουν έμφαση στην απλότητα, τη γενικότητα και τη χρηστικότητα στο Διαδίκτυο. Είναι μια μορφή δεδομένων κειμένου με ισχυρή υποστήριξη μέσω Unicode για διαφορετικές ανθρώπινες γλώσσες. Αν και ο σχεδιασμός της XML εστιάζει σε έγγραφα, η γλώσσα χρησιμοποιείται ευρέως για την αναπαράσταση αυθαίρετων δομών δεδομένων όπως αυτές που χρησιμοποιούνται σε υπηρεσίες web.

XML - χαρακτηριστικά

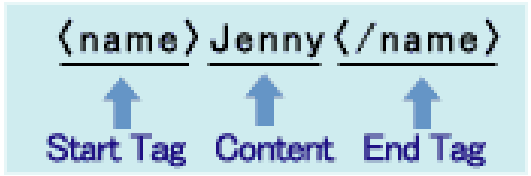
- Όταν περιγράφετε τη δομή και τη σημασία των δεδομένων σας, αποκτάτε τη δυνατότητα επαναχρησιμοποίησης αυτών των δεδομένων με διάφορους τρόπους.
- Μπορείτε να χρησιμοποιήσετε ένα σύστημα για να δημιουργήσετε τα δεδομένα και να τα επισημάνετε με ετικέτες XML και, στη συνέχεια, να επεξεργαστείτε αυτά τα δεδομένα σε οποιονδήποτε αριθμό από άλλα συστήματα, ανεξάρτητα από την πλατφόρμα υλικού ή το λειτουργικό σύστημα. Αυτή η δυνατότητα μεταφοράς είναι ο λόγος για τον οποίο η XML είναι σήμερα μία από τις πιο δημοφιλείς τεχνολογίες ανταλλαγής δεδομένων.

XML - Παράδειγμα

```
<?xml version="1.0"?>
<CAT>
  <NAME>Izzy</NAME>
  <BREED>Siamese</BREED>
  <AGE>6</AGE>
  <ALTERED>yes</ALTERED>
  <DECLAWED>no</DECLAWED>
  <LICENSE>Izz138bod</LICENSE>
  <OWNER>Colin Wilcox</OWNER>
</CAT>
```

XML – συνοπτικά:

- **Elements:** tags *<aTag>, <aTag/>* που μπορεί να περιέχουν data. Αποτελούνται από το "start tag," το "content tag," και το "end tag", π.χ.



Element χωρίς content: <element_name/> (χωρίς start και end tag)

- **Attributes:** περιέχουν πληροφορίες για τα elements, π.χ. <aTag *id="123"*>

Τα elements μπορεί να έχουν attributes, content καθώς και child elements.

XML vs HTML

- Τα έγγραφα HTML και XML περιέχουν δεδομένα που περικλείονται από ετικέτες, όμως, αυτή είναι και η μοναδική ομοιότητα μεταξύ των δύο γλωσσών. Στην HTML, οι ετικέτες ορίζουν την εμφάνιση και την αίσθηση των δεδομένων σας — οι τίτλοι τοποθετούνται εδώ, οι παράγραφοι ξεκινούν από εκεί και ούτω καθεξής. Στην XML οι ετικέτες ορίζουν τη δομή και τη σημασία των δεδομένων σας — ποια είναι τα δεδομένα.
- Δεν μπορείτε να χρησιμοποιήσετε HTML στη θέση της XML. Ωστόσο, μπορείτε να αναδιπλώσετε τα δεδομένα XML σε ετικέτες HTML και να τα εμφανίσετε σε μια ιστοσελίδα.
- Η HTML περιορίζεται σε ένα προκαθορισμένο σύνολο ετικετών που μοιράζονται όλοι οι χρήστες.
- Η XML σάς επιτρέπει να δημιουργήσετε οποιαδήποτε ετικέτα χρειάζεστε για να περιγράψετε τα δεδομένα σας και τη δομή αυτών των δεδομένων

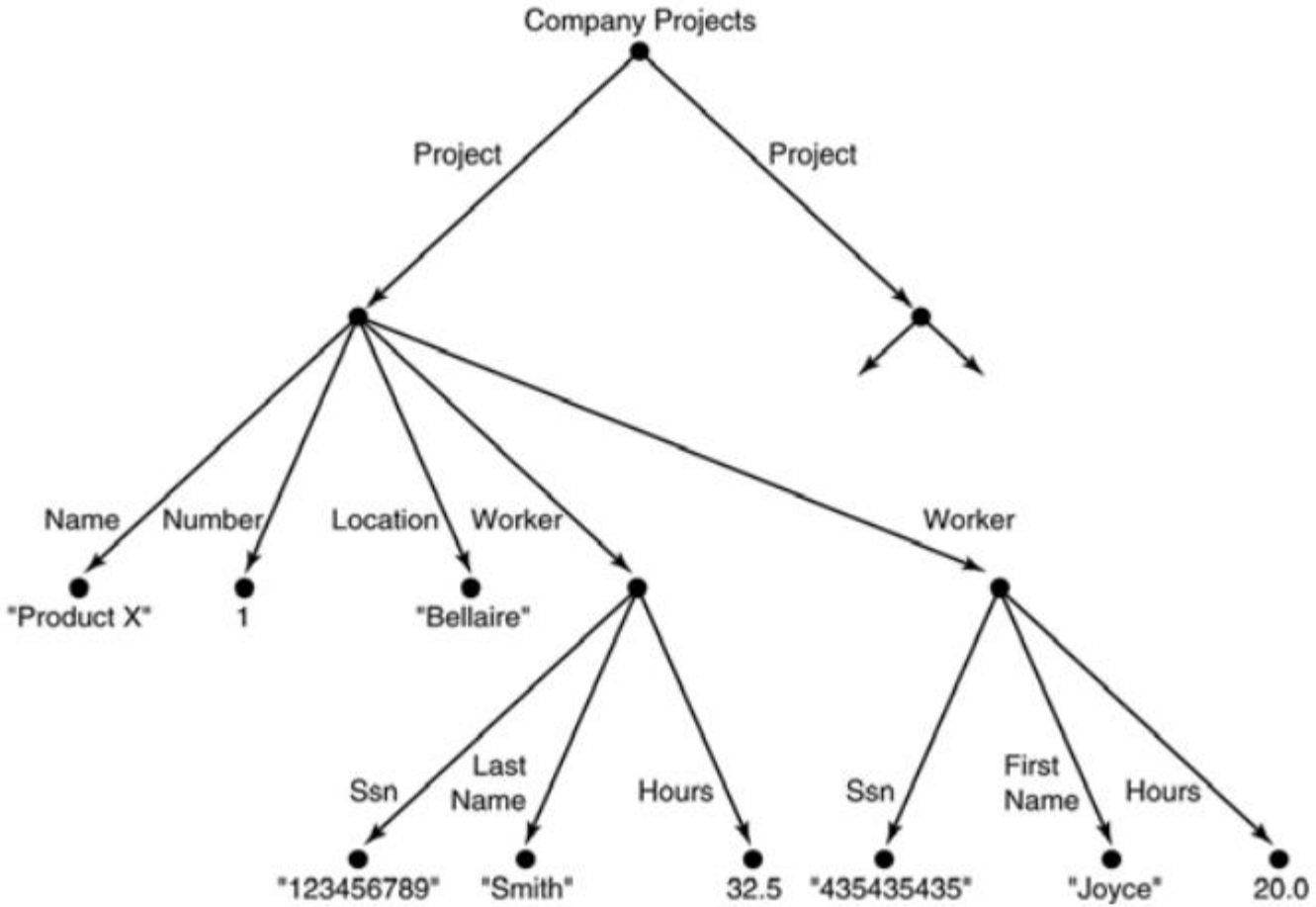
Ιεραρχική Δομή

```
<?xml version="1.0" standalone="yes"?>
<projects>

 <project>
  <Name>ProductX</Name>
  <Number>1</Number>
  <Location>Bellaire</Location>
  <DeptNo>5</DeptNo>
  <Worker>
   <SSN>123456789</SSN>
   <LastName>Smith</LastName>
   <hours>32.5</hours>
  </Worker>
  <Worker>
   <SSN>453453453</SSN>
   <FirstName>Joyce</FirstName>
   <hours>20.0</hours>
  </Worker>
 </project>
 </project>
  <Name>ProductY</Name>
  <Number>2</Number>
  <Location>Sugarland</Location>
  <DeptNo >5</DeptNo >
  <Worker>
   <SSN>123456789</SSN>
   <hours>7.5</hours>
  </Worker>
  <Worker>
   <SSN>453453453</SSN>
   <hours>20.0</hours>
  </Worker>
  <Worker>
   <SSN>333445555</SSN>
   <hours>10.0</hours>
  </Worker>
 </project>

...

</projects>
```



Ιεραρχική Δομή

Υπάρχουν δύο βασικές δομικές έννοιες: στοιχεία και γνωρίσματα. Τα γνωρίσματα παρέχουν επιπλέον πληροφορίες που περιγράφουν τα στοιχεία.

- Όπως και στην HTML, τα στοιχεία προσδιορίζονται σε ένα έγγραφο από την ετικέτα αρχής και την ετικέτα τέλους.

- Ετικέτα αρχής <name>

- Ετικέτα τέλους </name>

-Τα πολύπλοκα στοιχεία κατασκευάζονται από άλλα ιεραρχικά, ενώ τα απλά στοιχεία περιέχουν τιμές δεδομένων.

-Υπάρχει αντιστοιχία μεταξύ της XML και της δενδρικής δομής. Στη δενδρική δομή, οι εσωτερικοί κόμβοι παριστάνουν πολύπλοκα στοιχεία και οι κόμβοι φύλα παριστάνουν απλά στοιχεία. Γι’ αυτό το λόγο το μοντέλο της XML ονομάζεται δενδρικό ή ιεραρχικό.

Εφαρμογές XML

- Εφαρμογές
  - RSS
  - Atom
  - Office Open XML
  - OpenDocument
  - SVG
  - XHTML
- XML also provides the base language for communication protocols such as SOAP and XMPP.
- It is the message exchange format for the Asynchronous JavaScript and XML (AJAX) programming technique.

XML - ορολογία

Processor and application

The processor analyzes the markup and passes structured information to an application. The specification places requirements on what an XML processor must do and not do, but the application is outside its scope. The processor (as the specification calls it) is often referred to colloquially as an XML parser.

Markup and content

The characters making up an XML document are divided into markup and content, which may be distinguished by the application of simple syntactic rules. Generally, strings that constitute markup either begin with the character < and end with a >, or they begin with the character & and end with a ;. Strings of characters that are not markup are content. However, in a CDATA section, the delimiters <![CDATA[and]]> are classified as markup, while the text between them is classified as content. In addition, whitespace before and after the outermost element is classified as markup.

XML – ορολογία (2)

Character

An XML document is a string of characters. Every legal Unicode character (except Null) may appear in an (1.1) XML document (while some are discouraged).

Tag

A tag is a markup construct that begins with < and ends with >. There are three types of tag:

- start-tag, such as <section>;
- end-tag, such as </section>;
- empty-element tag, such as <line-break />.

Element

An element is a logical document component that either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag. The characters between the start-tag and end-tag, if any, are the element's content, and may contain markup, including other elements, which are called child elements. An example is <greeting>Hello, world!</greeting>. Another is <line-break />.

XML – ορολογία (3)

Attribute

An attribute is a markup construct consisting of a name–value pair that exists within a start-tag or empty-element tag. An example is , where the names of the attributes are "src" and "alt", and their values are "madonna.jpg" and "Madonna" respectively. Another example is <step number="3">Connect A to B.</step>, where the name of the attribute is "number" and its value is "3". An XML attribute can only have a single value and each attribute can appear at most once on each element. In the common situation where a list of multiple values is desired, this must be done by encoding the list into a well-formed XML attribute[i] with some format beyond what XML defines itself. Usually this is either a comma or semi-colon delimited list or, if the individual values are known not to contain spaces,[ii] a space-delimited list can be used. Welcome! , where the attribute "class" has both the value "inner greeting-box" and also indicates the two CSS class names "inner" and "greeting-box".

XML syntax

XML declaration

XML documents may begin with an XML declaration that describes some information about themselves. An example is <?xml version="1.0" encoding="UTF-8"?>.

Special characters

< represents "<";

> represents ">";

& represents "&";

' represents "'";

" represents “”.

Comments

Comments may appear anywhere in a document outside other markup. Comments cannot appear before the XML declaration. Comments begin with <!-- and end with -->. For compatibility with SGML, the string "--" (double-hyphen) is not allowed inside comments

Well-formed document

The XML specification defines an XML document as a well-formed text, meaning that it satisfies a list of syntax rules provided in the specification. Some key points in the fairly lengthy list include:

- The document contains only properly encoded legal Unicode characters.
- None of the special syntax characters such as < and & appear except when performing their markup-delineation roles.
- The start-tag, end-tag, and empty-element tag that delimit elements are correctly nested, with none missing and none overlapping.
- Tag names are case-sensitive; the start-tag and end-tag must match exactly.
- Tag names cannot contain any of the characters !"#$%&'()*+,/;<=>?@[\]^`{|}~, nor a space character, and cannot begin with "-", ".", or a numeric digit.
- A single root element contains all the other elements.

XML - example

Κείμενο 1:

```
<πρόσωπο>
 <όνομα>Διονύσης</όνομα>
 <ηλικία>5</ηλικία>
</πρόσωπο>
```

Κείμενο 2:

```
 <πρόσωπο όνομα="Διονύσης" ηλικία="5"/>
```

Κείμενο 3:

```
<πρόσωπο ηλικία="5">
        <όνομα>Διονύσης</όνομα>
</πρόσωπο>
```

XML schema

- Ως γλώσσα σήμανσης, η XML επισημαίνει, κατηγοριοποιεί και οργανώνει δομικά τις πληροφορίες. Οι ετικέτες XML αντιπροσωπεύουν τη δομή δεδομένων και περιέχουν μεταδεδομένα. Αυτό που υπάρχει μέσα στις ετικέτες είναι δεδομένα, κωδικοποιημένα με τον τρόπο που καθορίζει το πρότυπο XML. Ένα πρόσθετο σχήμα XML (XSD) ορίζει τα απαραίτητα μεταδεδομένα για την ερμηνεία και την επικύρωση της XML (Αυτό αναφέρεται επίσης ως το κανονικό σχήμα). Ένα έγγραφο XML που συμμορφώνεται με βασικούς κανόνες XML είναι "καλά διαμορφωμένο» (well-formed) και ένα που τηρεί το σχήμα του είναι "έγκυρο“ (valid).

XML Schema

- Ένα σχήμα XML είναι μια γλώσσα για την έκφραση περιορισμών σχετικά με έγγραφα XML.
- Document Type Definitions (DTDs)
- W3C XSD (XML Schema Definitions)

XML Schema

Ένα σχήμα μπορεί να χρησιμοποιηθεί :

- Για να παρέχει μια λίστα στοιχείων και χαρακτηριστικών σε ένα λεξιλόγιο.
- για να συσχετίσετε τύπους, όπως ακέραιος αριθμός, συμβολοσειρά κ.λπ., ή πιο συγκεκριμένα, όπως hatsize, sock_colour, κ.λπ., με τιμές που βρίσκονται σε έγγραφα.
- Για να περιορίσετε πού μπορούν να εμφανίζονται στοιχεία και χαρακτηριστικά και τι μπορεί να εμφανίζεται μέσα σε αυτά τα στοιχεία, όπως π.χ. ότι ένας τίτλος κεφαλαίου εμφανίζεται μέσα σε ένα κεφάλαιο και ότι ένα κεφάλαιο πρέπει να αποτελείται από έναν τίτλο κεφαλαίου ακολουθούμενο από μία ή περισσότερες παραγράφους κειμένου.
- Για να παρέχει τεκμηρίωση που είναι τόσο αναγνώσιμη από τον άνθρωπο όσο και μηχανική επεξεργασία·
- Για να δώσει μια επίσημη περιγραφή ενός ή περισσότερων εγγράφων.

DTD declaration

- Elements:

<!ELEMENT html (head, body)>
<!ELEMENT p (#PCDATA | p | ul | dl | table | h1|h2|h3)*>

(#PCDATA): historically meaning parsed character data, this means that only one text element is allowed

«,»: λίστα με συγκεκριμένη σειρά

«|»: λίστα με αμοιβαία αποκλειόμενα στοιχεία

Ποσοδείκτες (quantifiers)

«+»: 1 ή περισσότερες εμφανίσεις

«*»: 0 ή περισσότερες εμφανίσεις (προαιρετικό)

«?¨»: το πολύ 1 εμφάνιση (προαιρετικό)

Χωρίς ποσοδείκτη: ακριβώς μια φορά (υποχρεωτικά)

DTD declaration

- Attributes:

```
<!ATTLIST img
   src    CDATA        #REQUIRED
   id     ID           #IMPLIED
   sort   CDATA        #FIXED "true"
   print  (yes | no) "yes"
>
```

CDATA: this type means characters data and indicates that the effective value of the attribute can be any textual value

ID: the effective value of the attribute must be a valid identifier

DTD – Παράδειγμα 1A

Αρχείο «example.dtd»

```
<!ELEMENT people_list (person)*>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber (#PCDATA)>
```

DTD – Παράδειγμα 1Β

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

DTD – Παράδειγμα 1Γ

Ή σε ένα αρχείο:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE people_list [
  <!ELEMENT people_list (person*)>
  <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT birthdate (#PCDATA)>
  <!ELEMENT gender (#PCDATA)>
  <!ELEMENT socialsecuritynumber (#PCDATA)>
]>
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

XML Schema

Τα XSD είναι πολύ πιο ισχυρά από τα DTD στην περιγραφή γλωσσών XML. Χρησιμοποιούν ένα πλούσιο σύστημα τυποποίησης δεδομένων και επιτρέπουν πιο λεπτομερείς περιορισμούς στη λογική δομή ενός εγγράφου XML. Τα XSD χρησιμοποιούν επίσης μια μορφή που βασίζεται σε XML, η οποία καθιστά δυνατή τη χρήση συνηθισμένων εργαλείων XML για τη διεκπεραίωση τους.

Το στοιχείο xs:schema ορίζει ένα schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"></xs:schema>
```

DTD - πλεονεκτήματα

- Η υποστήριξη DTD είναι πανταχού παρούσα λόγω της συμπερίληψής της στο πρότυπο XML 1.0.
- Τα DTD είναι λακωνικά σε σύγκριση με γλώσσες σχήματος που βασίζονται σε στοιχεία και κατά συνέπεια παρουσιάζουν περισσότερες πληροφορίες σε μία μόνο οθόνη.
- Τα DTD επιτρέπουν τη δήλωση τυπικών συνόλων δημόσιων οντοτήτων για δημοσίευση χαρακτήρων.
- Τα DTD ορίζουν έναν τύπο εγγράφου αντί για τους τύπους που χρησιμοποιούνται από έναν χώρο ονομάτων, ομαδοποιώντας έτσι όλους τους περιορισμούς για ένα έγγραφο σε μια ενιαία συλλογή.

DTD - περιορισμοί

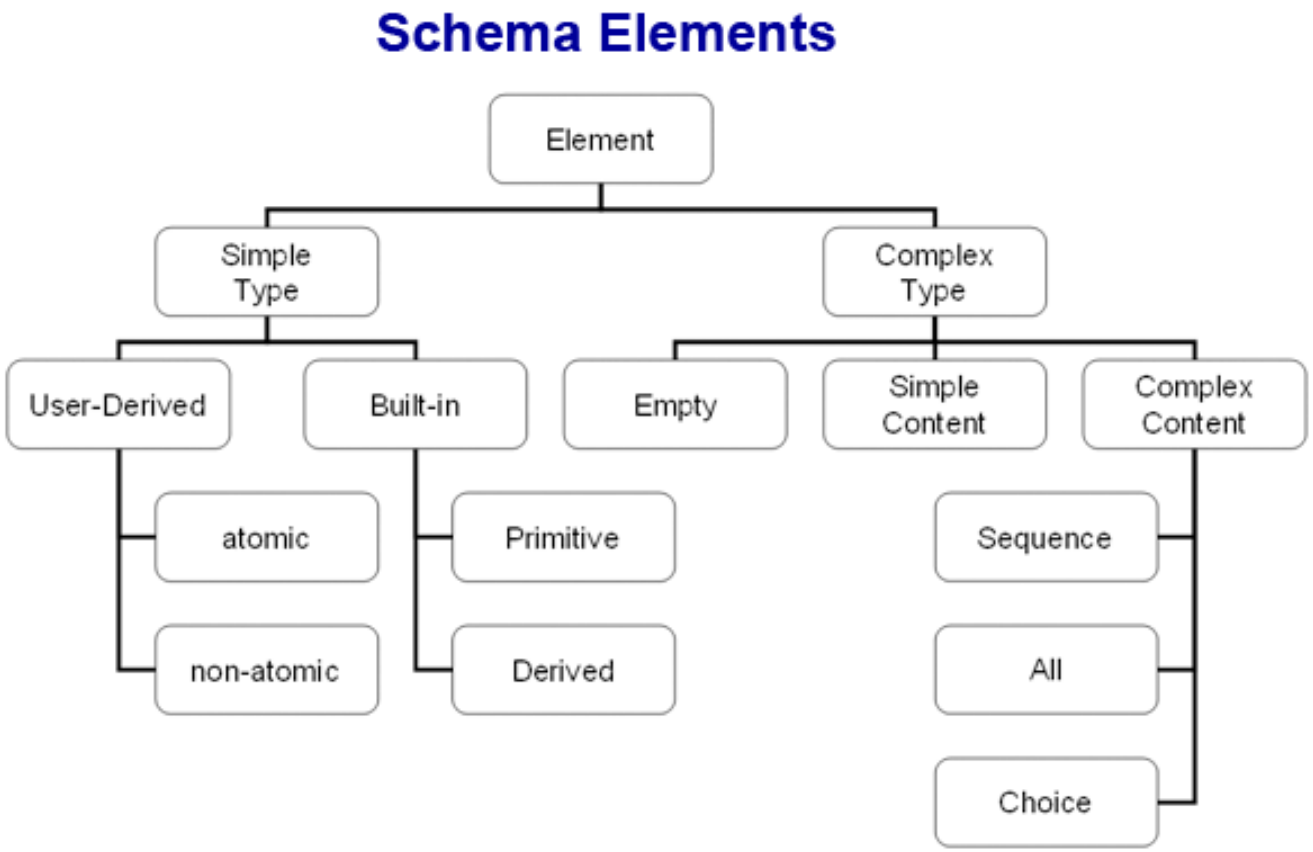
- Δεν έχουν ρητή υποστήριξη για νεότερες δυνατότητες της XML, κυρίως για χώρους ονομάτων.
- Τους λείπει η εκφραστικότητα. Τα XML DTD είναι απλούστερα από τα SGML DTD και υπάρχουν ορισμένες δομές που δεν μπορούν να εκφραστούν με κανονικές γραμματικές. Τα DTD υποστηρίζουν μόνο στοιχειώδεις τύπους δεδομένων.
- Δεν έχουν αναγνωσιμότητα. Οι σχεδιαστές DTD συνήθως κάνουν μεγάλη χρήση οντοτήτων παραμέτρων (οι οποίες συμπεριφέρονται ουσιαστικά ως μακροεντολές κειμένου), οι οποίες διευκολύνουν τον ορισμό σύνθετων γραμματικών, αλλά σε βάρος της σαφήνειας.
- Χρησιμοποιούν μια σύνταξη βασισμένη σε σύνταξη κανονικής έκφρασης, που κληρονομήθηκε από το SGML, για να περιγράψουν το σχήμα. Τυπικά API XML, όπως το SAX, δεν προσπαθούν να προσφέρουν στις εφαρμογές μια δομημένη αναπαράσταση της σύνταξης, επομένως είναι λιγότερο προσιτή στους προγραμματιστές από ό,τι μια σύνταξη που βασίζεται σε στοιχεία.

XML Schema - example

```
<xsd:element name="CAT">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="NAME" type="xsd:string"/>
      <xsd:element name="BREED" type="xsd:string"/>
      <xsd:element name="AGE" type="xsd:positiveInteger"/>
      <xsd:element name="ALTERED" type="xsd:boolean"/>
      <xsd:element name="DECLAWED" type="xsd:boolean"/>
      <xsd:element name="LICENSE" type="xsd:string"/>
      <xsd:element name="OWNER" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

XML Schema

- Ένα element σύνθετου τύπου μπορεί να περιέχει child elements και attributes, ενώ ένα element απλού τύπου μπορεί να περιέχει μόνο text.



XML Schema

XSD provides a set of 19 primitive data types (anyURI, base64Binary, boolean, date, dateTime, decimal, double, duration, float, hexBinary, gDay, gMonth, gMonthDay, gYear, gYearMonth, NOTATION, QName, string, and time). It allows new data types to be constructed from these primitives by three mechanisms:

restriction (reducing the set of permitted values),

list (allowing a sequence of values), and

union (allowing a choice of values from several types).

Παράδειγμα – Κτηματολόγιο – XML έγγραφο

Έστω το ακόλουθο απόσπασμα ενός κειμένου XML μιας κτηματολογικής εφαρμογής, που περιγράφει ένα γεωτεμάχιο (με κωδικό «Γ123κ», δικαιούχο την Μαρία Παράσχου και εμβαδόν 1200 m2)

```
<γεωτεμάχιο id= “Γ123κ”>
        <δικαιούχος>Μαρία Παράσχου</δικαιούχος>
        <εμβαδόν>1200</εμβαδόν>
</γεωτεμάχιο>
```

Παράδειγμα – Κτηματολόγιο με DTD

<!ELEMENT γεωτεμάχιο (δικαιούχος, εμβαδόν)>

<!ATTLIST γεωτεμάχιο id CDATA>

<!ELEMENT δικαιούχος (#PCDATA)>

<!ELEMENT εμβαδόν (#PCDATA)>

Παράδειγμα – Κτηματολόγιο με XSD:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="γεωτεμάχιο">
                <complexType>
                        <sequence>
                                <element name="δικαιούχος" type="string"/>
                                <element name="εμβαδόν " type="unsignedInt"/>
                        </sequence>
                        <attribute name="id">
                                <simpleType>
                                        <restriction base="string">
                                                <pattern value="Γ\d{3}[A-Za-z]{1}"/>
                                        </restriction>
                                </simpleType>
                        </attribute>
                </complexType>
        </element>
</schema>
```

Παράδειγμα 2 – χρήση id, idref

```
<οικογένεια>
        <πρόσωπο id="ο55">
                <όνομα>Ιωάννα</όνομα>
        </πρόσωπο>
        <πρόσωπο id="ο44">
                <όνομα>Παντελής</όνομα>
                <παιδί_της idref="ο55"/>
        </πρόσωπο>
 ...
</οικογένεια>
```

Παράδειγμα

- Βιβλιοπωλείο
- Βιβλίο: τίτλος, έτος έκδοσης, εκδότης, συγγραφέας (όνομα, επώνυμο, εθνικότητα)
- Σχήμα με DTD, με XSD, έγγραφο με παραδείγματα

Βιβλιοπωλείο – 1η προσέγγιση – Παράδειγμα δεδομένων

```
<books>
        <book>
                <title>1984</title>
                <author>George Orwell</author>
                <language>English</language>
                <year>1949</year>
                <publisher>ABC</publisher>
        <book>
<books>
```

Βιβλιοπωλείο – 1η προσέγγιση - DTD

<!ELEMENT books(book*)>

<!ELEMENT book(title, author, language, year?, publisher)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT author(#PCDATA)>

<!ELEMENT language(#PCDATA)>

<!ELEMENT year (#PCDATA)>

<!ELEMENT publisher (#PCDATA)>

Παραδοχές:
Σε κάποιο βιβλίο π.χ. αρχαίο κείμενο, δεν υπάρχει έτος

Βιβλιοπωλείο – 1η προσέγγιση - XSD

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="books">
                <complexType>
                        <sequence>
                                <element name="book”>
                                        <complexType>
                                                <sequence>
                                                        <element name="title" type="string"/>
                                                        <element name="year" type="unsignedInt"/>
                                                        <element name="author" type="string"/>
                                                        <element name="language" type="string"/>
                                                        <element name="publisher" type="string"/>
                                                </sequence>
                                        </complexType>
                                </element>
                        </sequence>
                </complexType>
        </element>
</schema>
```

Βιβλιοπωλείο – 2η προσέγγιση - XSD

- Αν θέλουμε να προσθέσουμε την «Φάρμα των ζώων» του ίδιου συγγραφέα, τότε έχουμε μια επανάληψη πληροφορίας σχετικά με τον συγγραφέα (όνομα, γλώσσα και αν είχαμε επιπλέον πληροφορίες, π.χ. βιογραφικό)

- → Συγγραφέας ως ξεχωριστή οντότητα

Βιβλιοπωλείο – 2η προσέγγιση – Παράδειγμα δεδομένων

```
<bookstore>
        <books>
                <book author_id="1">
                        <title>1984</title>
                        <year>1949</year>
                        <publisher>ABC</publisher>
                </book>
        </books>
        <authors>
                <author id="1">
                        <name>George Orwell</name>
                        <language>English</language>
                </author>
        </authors>
</bookstore>
```

Χρειαζόμαστε ένα νέο root στοιχείο, δεν μπορούμε πλέον να ξεκινάμε με books, αφού ακολουθεί και το authors

Βιβλιοπωλείο – 2η προσέγγιση - DTD

<!ELEMENT bookstore(books, authors)>

<!ELEMENT books(book*)>

<!ELEMENT book(title, year?, publisher)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT year (#PCDATA)>

<!ELEMENT publisher (#PCDATA)>

<!ELEMENT authors(author*)>

<!ELEMENT author(name, language)>

<!ELEMENT name(#PCDATA)>

<!ELEMENT language(#PCDATA)>

<!ATTLIST author id ID #REQUIRED>

<!ATTLIST book author_id IDREF #REQUIRED>

Βιβλιοπωλείο – 2η προσέγγιση - XSD

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
 <element name="bookstore">
   <element name="books">
                  <complexType>
                   <sequence>
       <element name="book">
        <complexType>
         <sequence>
          <attribute name="author_id" />
          <element name="title " type="string"/>
          <element name="year " type="unsignedInt"/>
          <element name="publisher" type="string"/>
         </sequence>
        </complexType>
       </element>
                                 </sequence>
    </complexType>
   </element>
```

```
<element name="authors">
                                  <complexType>
                        <sequence>
       <element name="author">
        <complexType>
         <sequence>
          <attribute name="id" />
          <element name="name" type="string"/>
          <element name="language" type="string"/>
         </sequence>
        </complexType>
       </element>
                                  </sequence>
    </complexType>
   </element>
                 </element>
</schema>
```