Universidad Politécnica de Cartagena

Applications on the Internet

Lesson 3. Structured documents and markup languages

Goals

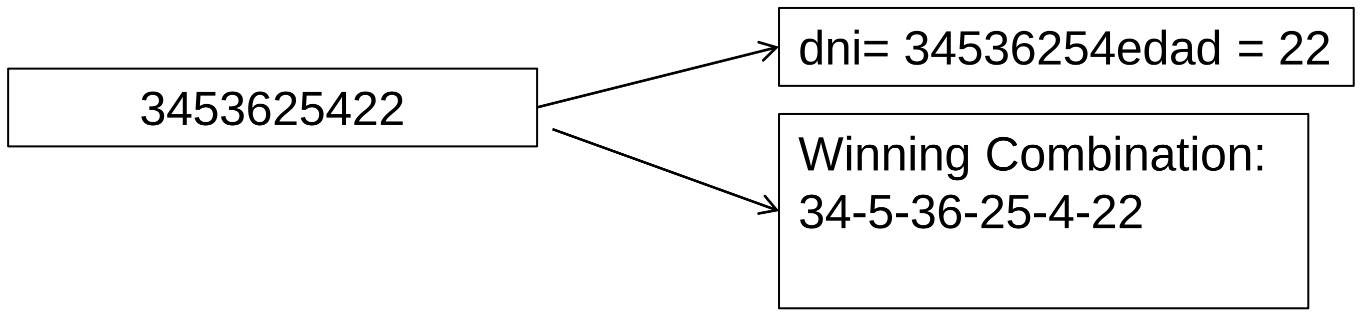
- *Understand the usefulness of markup languages and metalanguages to structure the information / data in the context of the Web.*
- *Recognize the need and usefulness of design templates.*
- *Understand the need for a neutral API for processing of structured documents.*
- *Develop HTML documents with multiple elements.*
- *Develop and apply design templates with CSS.*
- *Understand an HTML document and the function of the different elements (marks).*

Contents

Introduction

Data and structure

- Information (data, content, ..) is accompanied by structure (**= organization**) usually.
- This structure can be **explicit or implied**. Humans are able to extract the structure from the context. For machines it is much more difficult.
- The structure (organization) may be logical, hierarchical, semantic ...



Introduction

A markup language is a vocabulary (set of tags) that allows to annotate data in a document.

Markup languages allow you to structure the contents of a document.

- The document will include content and notation/tags.
- The tags provide *context to the content*. That is, provide **metadata:** "Data about data".
- **Types** of **tags**:
  - For **presentation**or format: describe how to represent data. Example: word processing
  - For **processing**: Describe how to process the data. Example: word processing (latex).
  - **Descriptive** or semantic: label the meaning of the data.
- These categories are blurred in many systems, but normally is preferably a clear separation.

contents

SGML

The use of tags allows us to provide context or meaning to the content of a document.

But when I examine a marked document, how do I know which tags I can use? where are they declared? When can I use them and how? What are the relationships between tags?

I need to provide a "manual" for tags.

- This can be done simply by a specification of the document in natural language.

But a machine is not able to understand a natural language specification.

- A **formal declaration** is needed: A description of the tags that is capable of being analyzed and understood by a program.
- A formal declaration allows to **check the validity of the marking** of a document by an appropriate program

DTD and SGML

Document Type Definition (DTD): **It is a formal declaration of a type of markup language, the tags available, the relationship between them and their meaning.**

- The DTD is the template upon which the marked content can be understood, interpreted and validated **automatically.**

Standard Generalized Markup Language (SGML): **It provides a formal syntax for declaring DTDs.**

- ISO standard (ISO 8879: 1986 SGML).
- A validator program receives a DTD and a document and check whether the marks were used in a manner consistent with his specification.

From SGML various markup languages are declared.

- Among them, the most commonly used: HTML, up to version 4.0.
- However, HTML 5 has not been declared by a formal DTD, but in natural language. HTML5 is a vocabulary and a set of APIs associated.

SGML is too complex and ambiguous in some respects: XMLwas developed to solve some of these problems

contents

Introduction HTML-

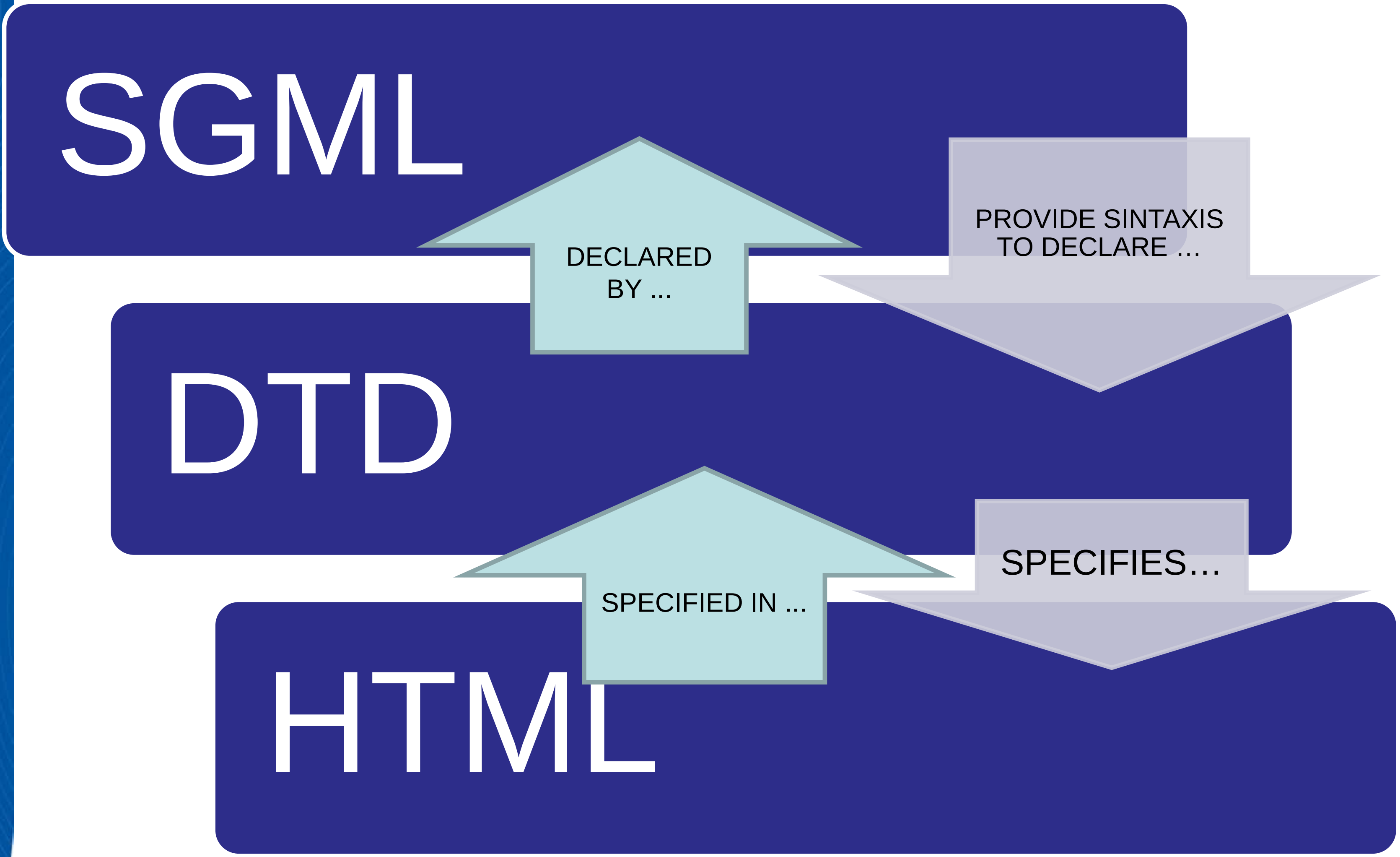
HyperText Markup Language (HTML): markup language used for representing resources on the Web.

- It is the language for publishing content on the Web.
- Originally developed by Tim Berners-Read.
- Its main original goal was to provide a simple means of displaying information that could be read by the user.
- HTML did not provide (from the outset) all necessary generality and has been (and is) subject to revision.
- It is constantly evolving and in extension. As new formats are introduced into the web, it extends its functionality.

http://www.w3.org/TR/html4/sgml/dtd.html

  - DTD version 4.

Relationship HTML-DTD-SGML



HTML -Elements

HTML tags are called *elements.*

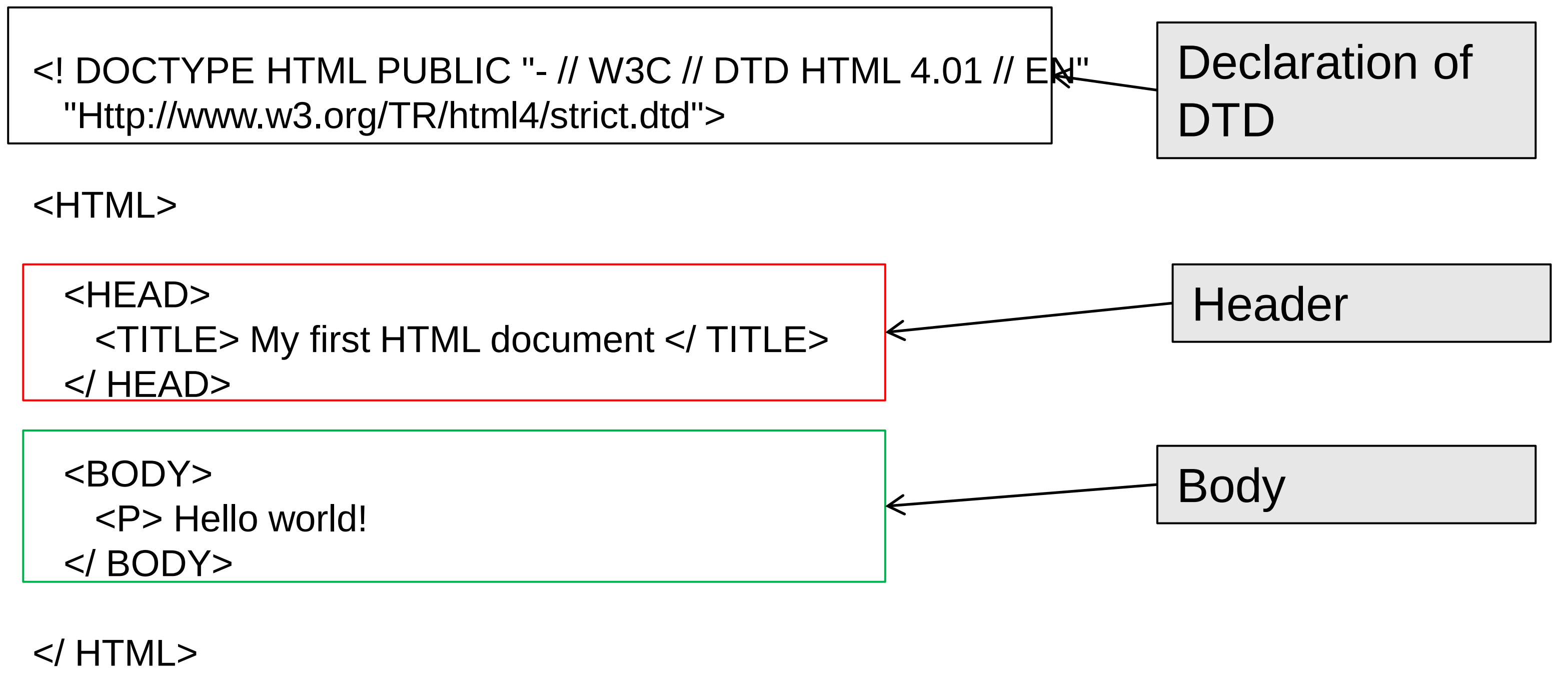
- An element consists of a start-tag and end-tag and contains
- Some items have no end tag.
- The **start tag** includes the element name in "<>". The **end tag** includes the element name between "</>"
  - Example: <h1> </ h1>
- Start tags may contain **attributes**, which are **key = value pairs**.
  - Example: <a href= "Index.html">
- The **elements are nested**. An element may contain other elements.
  - Example: <ul> <li>Item 1 </li> <ul>
- **Tree structure.**

HTML - structure

An HTML document is a text document.

It contains three parts:

- **Declaration** information
- **Header** with information about the document:
- The **body** of the document containing the content itself.
- The documents begin and end with the <html> and </html> tags
- You can use comments between <! - and ->



```
<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.01 // EN"
  "Http://www.w3.org/TR/html4/strict.dtd">

<HTML>

  <HEAD>
    <TITLE> My first HTML document </ TITLE>
  </ HEAD>

  <BODY>
    <P> Hello world!
  </ BODY>

</ HTML>
```

HTML- Main elements in HTML 4

- Headers: *H1, H2, H3, H4, H5, H6*
- Lines and paragraphs: *P* Y *BR*
- links: *A* in the body and *LINK* in the header
- Images: *IMG*
- lists
  - List item: *LI*
  - Unordered: *UL*
  - Ordered: *OL*
  - Definition: *DL, DT, DD*
- Tables: *TABLE, TR, TH, TD*
- Objects and applets: *OBJECT, APPLET*
- Forms and scripts: *FORM, SCRIPT*
- Group: *DIV*, *SPAN*

HTML- Main elements in HTML 4

Element identifiers: are attributes that can be used on any item and allow identification

- *id:* its value must be **unique** in a document
- *class:* assigns a class name, so multiple elements can share the same *class* value.

Grouping elements: *DIV and SPAN*

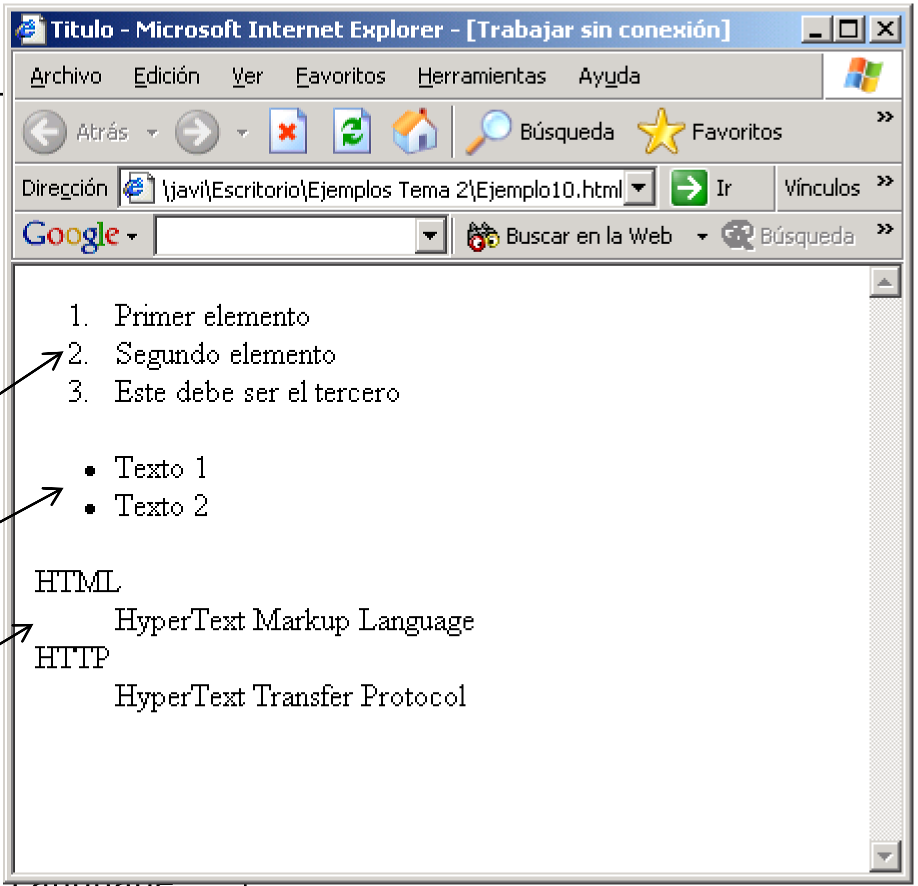
- Allow you to group another set of elements. They are very useful in conjunction with style sheet (CSS) properties.

```
<DIV class = "section" id = "forest-elephants" >
        <H1> Forest elephants </ H1>
        <P class = "big"> In this section, we discuss the Lesser Known
forest elephants. ... this section continues ...
        <DIV class = "subsection" id = "forest-habitat" >
                <H2> Habitat </ H2>
                <P class = "big"> Forest elephants do not live in trees
but Among them. ... this subsection continues ...
        </ DIV>
</ DIV>
```

Examples of HTML-elements

lists

```
<Html>
<! - Example 10 ->

<Head>
  <Title>Title</ Title>
</ Head>

<Body>
  <P>
  <ol>
    <li> First element
    <li> Second element
    <li> East should be the third
  </ol>
  </ P>

  <ul>
    <li> Text 1
    <li> Text 2
  </ul>

  <P>
  <Dl>
    <dt> HTML <dd> HyperText Markup Language
    <dt> HTTP <dd> HyperText Transfer Protocol
  </ Dl>
  </ P>
</ Body>

</ Html>
```


Titulo - Microsoft Internet Explorer - [Trabajar sin conexión]
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección \javi\Escritorio\Ejemplos Tema 2\Ejemplo10.html Ir Vínculos
Google Buscar en la Web Búsqueda

1. Primer elemento
2. Segundo elemento
3. Este debe ser el tercero

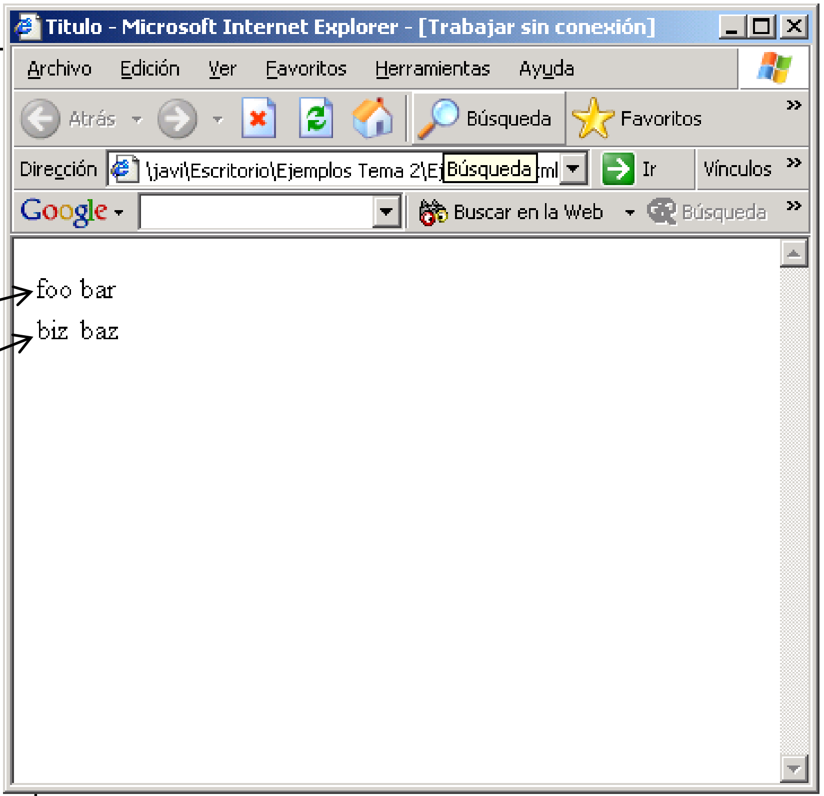
- Texto 1
- Texto 2

HTML
    HyperText Markup Language
HTTP
    HyperText Transfer Protocol

Examples of HTML-elements

Boards

```
<Html>
<! - Example 14 ->

<Head>
  <Title> Title </ title>
</ Head>

<Body>
  <Table>
    <Tr>
      <Td> foo </ td> <td> bar </ td>
    </ Tr>
    <Tr>
      <Td> biz </ td> <td> baz </ td>
    </ Tr>
  </ Table>
</ Body>
</ Html>
```


Titulo - Microsoft Internet Explorer - [Trabajar sin conexión]
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás
Búsqueda
Favoritos
Dirección
\javi\Escritorio\Ejemplos Tema 2\E
Ir
Vínculos
Google
Buscar en la Web
Búsqueda

```
foo bar
biz baz
```

Activity 1. Practice with HTML elements

- http://ait.upct.es/asignaturas/ai/html4.html
  - Examine the complete source code of the document.
  - Examine the code with the inspector in your browser. In most cases, it is faster to search for items of interest with the inspector.
  - List the elements used on this page as previously described.
- www.w3schools.com/
  - Examine the tutorials and examples of the various elements, in particular, tables and lists.
  - This website provides an online HTML editor that is very useful to practice. Use the editor "Try It“ for any example.
- Replace the sample code in "Try it“ by the the source code of http://ait.upct.es/asignaturas/ai/html4.html and try it.
  - Verify that certain images are not displayed, why ?, what is shown instead?
  - Modify the code to display images which are not shown now, how can you do this?

Evolution of HTML

- HTML 1 (Berners Lee, 1989): very basic, limited integration of multimedia. In 1993, Mosaic added many new features (eg, embedded images).
- HTML 2.0 (IETF, 1994) attempted to standardize these and other properties, but later, between 1994-96, Netscape and IE added many new (and divergent) functionalities.
- HTML 3.2 (W3C, 1996) tried to unify into a single HTML standard, but again encountered unforeseen types of content, such as Java applets and video streams.
- HTML 4.0 (W3C, 1997): current standard, is designed to anticipate future developments.
- XHTML 1.0 (W3C, 2000): 4.01 HTML version modified to be compatible with XML.
- HTML 5 (W3C, 2004) current standard, replaces the previous one and introduces new tags.

Limitations and evolution of HTML

HTML 3.2

- His main **problem** is that used both **descriptive** tags as well as tags for **attributes** and **formatting**.
  - Example: <body bgcolor= "Red">
  - Difficult to update and insert content.
  - Hinders the selection of a particular format, and especially, its change. Sometimes (eg. corporate portal) all pages must adopt the same format: with HTML 3.2 this is tedious and requires the modification of all the documents.
  - By itself, HTML does not have the flexibility to specify formats.

Limitations and evolution of HTML

HTML 4.0

- HTML 4.0 tried to **separate the "content" from the "presentation“:** using a generalized mechanism of **style sheets (CSS)**.
- A consistent mechanism is provided to insert embedded objects (such as a Java applet or Flash animation).
- You can use *script languages* in a more flexible and useful way.
- There are alternate versions
  - **Strict**: It does not provide formatting tags.
  - **In transition**: Includes format tags

Limitations and evolution of HTML

Lack of interactivity

- One of the main problems of HTML from its inception was the lack of mechanisms to interact with the user: a static HTML document is not changed due to user actions (such as moving the mouse).
- This prevents the development of interactive content and complex applications for the web.

Solution: send code to documents to be executed by the client (ie browser).

- *Javascript* is the standard "light" language and has become a fundamental part of HTML 5.
- Another option is to run code through *plugins* (such as Java applets). Deprecated

Limitations and evolution of HTML

HTML 5.0

- A new version of the languaje with **new marks**.
- Does not use a formal declaration (DTD).
- Declares a set of APIs to process and interact with HTML documents.
- Does not require to implement the APIs in javascript, but it is the most common case in browsers

contents

Stylesheets

Cascading style Sheets (CSS)

- Style Description Language to specify the format of data of structured documents.
- The objective is the **separation of content and presentation**: Data are independent of format, platform, vendor and device.
- Simplify the development of HTML pages (and other structured documents such as XML documents) and facilitate maintenance.
- Improves transmission efficiency:
  - Style information is sent only once, the browser caches it.
- You can define a style sheet for each media destination: graphical browser, printer, mobile, voice browser, braille browser, etc.
  - Also as a function of certain characteristics of the device, eg. resolution
  - http://www.w3.org/TR/css3-mediaqueries/

Declaration CSS-

A CSS style sheet is a text document containing a number of format attributes to apply to the elements of a structured document associated.

- Each rule (selector + declarations) determines the properties of the element:

SELECTOR {property_1: Value_1;

...

property_2: Value_N; }

What value have the properties?

To what elements it affects?

H1 {color: green; }

CSS - Properties

With CSS you can assign values to hundreds of formatting properties

- Fonts, colors, margins, alignment, shading, background images, etc.
- Furthermore, **dynamic properties, in response to certain events,** can be established. For example, it can change the color of an item when the mouse passes over it.
- CSS specification defines algorithms to display items on the screen, which are implemented by browsers. It is essential to understand the **positioning model**.

CSS - Insertion

Style sheets can be associated with a document in 3 ways:

- **Internal** to a document
  - <STYLE type= "text/css"> Rules </ STYLE>
- **External** to a document
  - <LINK rel= "stylesheet" href= "URL" type= "text/css">
- As an **attribute of** an element
  - <P style= "font-size: 20pt "> text </ P>
- Obviously, to obtain the greatest benefit of the CSS it is preferable to associate an external document.
- The association embedded in the document (internal and attribute) is useful when javascript is used to dynamically modify the format of a document, as a result of user actions.

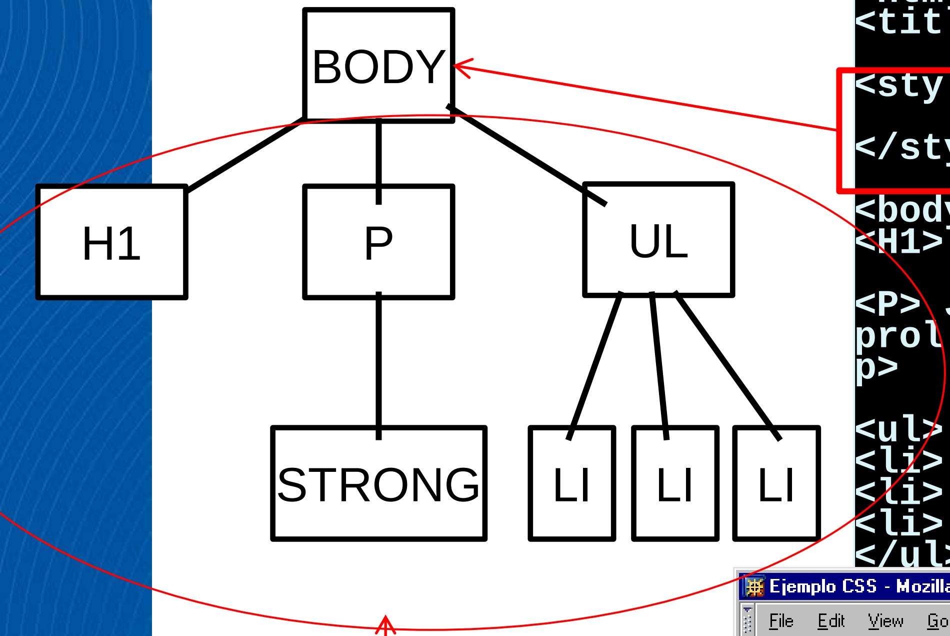
CSS - Inheritance

CSS are called "cascade" since child elements inherit default style values from parents.

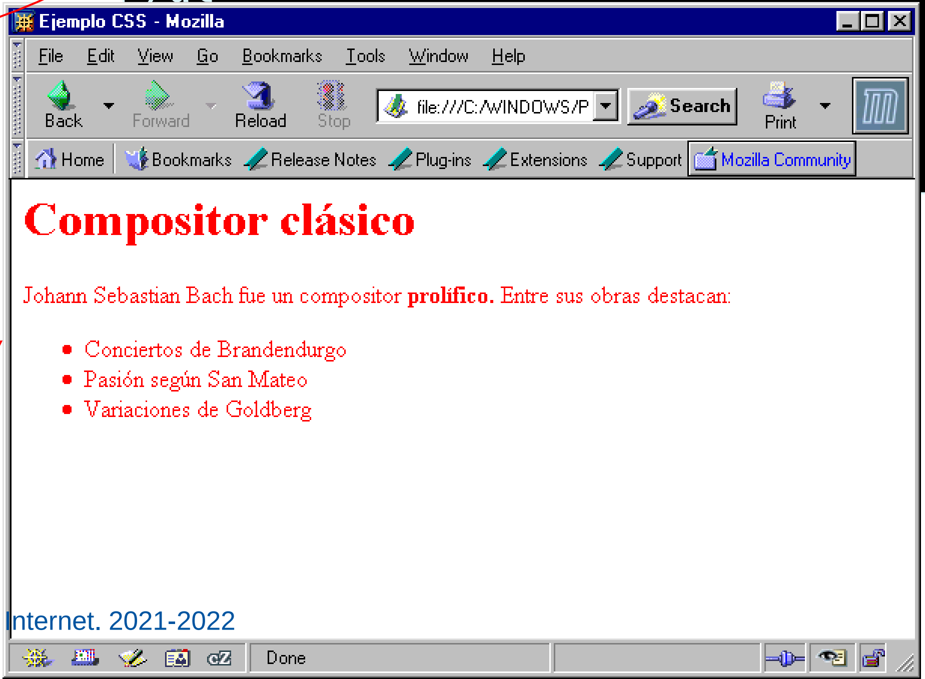
The values of the properties of an element are assigned by specific, inherited or default ones (from highest to lowest preference, respectively).

- More specific selectors overwrite the more general
- The rules apply according to specificity, not the order they appear in the document. If repeated, the last occurrence is chosen.

CSS - Inheritance



```
<html>
<title> Example CSS </title>

<style>
        body {Color: red}
</style>

<body>
<H1> Composer cl & aacute; sico </ H1>

<P> Johann Sebastian Bach was a composer <strong
prol RIVER fico. </strong> His works include: </
p>

<ul>
<li> Concerts tagendurgo
<li> St. Matthew Passion
<li> Variations Goldberg
</ul>
```


Ejemplo CSS - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop file:///C:/WINDOWS/P Search Print
Home Bookmarks Release Notes Plug-ins Extensions Support Mozilla Community
Done

Compositor clásico

Johann Sebastian Bach fue un compositor **prolífico.** Entre sus obras destacan:

- Conciertos de Brandendurgo
- Pasión según San Mateo
- Variaciones de Goldberg

All elements inherit the color property

CSS - Selectors

Selectors are patterns of selection rules that apply to an item

Selector switch	Meaning	Example
*	Any element	* {font-family: Arial}

E	Any E element	LI {font-family: Arial}
		P {color: black}

E> F	Any element F *son* E	UL> LI {font-size: 70%}
		OL> LI {font-size: 115%}

E F	Any element F *descendant* E	MS H1 {color: blue}
E + F	F immediately after E (at the same level of the tree)	MS + H1 {color: blue}
E[foo= "A"]	Any element E with the attribute foo equal to "a"	IMG [src= "Logo.gif"] {width: 100px}

CSS - Classes, ID and pseudoclasses

The selection of items using patterns can be very complex

- In many cases it is easier to "name" elements to apply a certain style.
- the attribute **“id“** is used to **univocally identify** an element (all ids must be unique in a document).
- the attribute **"class“** is used to identify a **set of elements** with shared attributes.
- Sometimes more flexibility is needed. **Pseudo classes and pseudo elements** are defined: identified with ***":“***. For example: :first-child, :visited, :link, :hover, :active, :focus

CSS - Examples of classes, id and pseudoclaes

Selector with classes and identifiers

- **E # myid** E element whose "id" attribute is "myid"

  TABLE # Prices {text-align: Center}

- **E.myid** Any element E whose attribute "class" is "myid"

  TABLE.precios {text-align: Center}

- The pseudoclasses can be applied to any other selector and combined.

  <A class= "external" href= “http://out.side/"> external link </A>

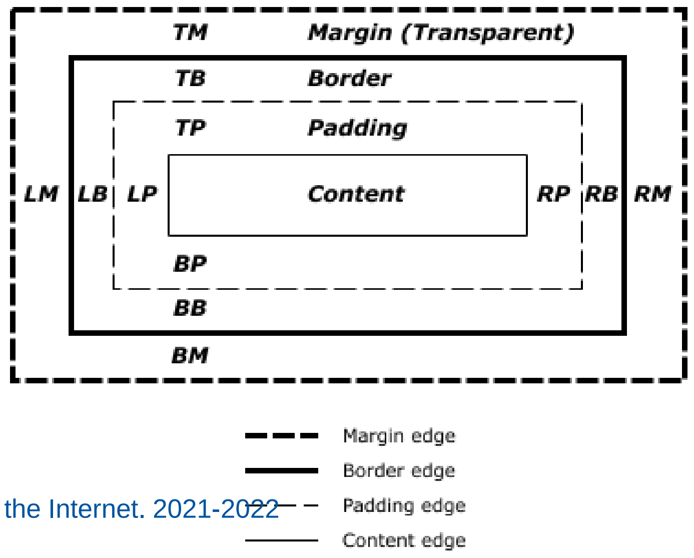
  a.external: visited { color: blue }

  li: hover {font-size: 14em}

CSS - Box model

CSS also defines how elements are displayed

- **Box model**: Each HTML element is associated with a rectangular box (remember: elements include tags and what they contain).
- The box of an element that contains other elements, will contain the child elements. But it depends on the visual formatting model (below).
- We can define CSS properties for each of the areas
- the properties *width* and *height* refer only to the content of the box.



CSS - Boxes and representation

Elements *inline* and elements *block*

- The elements are assigned a default value: *block* or *inline*
- The width and height of a box can only be specified for *block* elements. In the *inline* elements they are adjusted according to the parent they are contained in

Property *display*

- This allows to change the type of box assigned to an element (*inline* or *block*), which modifies its representation (though not the kind of element that is). It can also be *inline-block:* lets you specify width and height, ie, displays inline as if they were *block*.
- Show or hide items:
  - *hidden*: The item is not displayed but continues to occupy space
  - *none*: The item is not displayed and does not takes up space. For all purposes it's like removed from the tree.

CSS - visual formatting model

CSS specifies how the elements are positioned on the screen

- Each element of the tree generates zero or more boxes according to the box model. Boxes of child elements will be contained by the parent element.
- The positioning depends on the dimensions of the box, and the relationships between the elements and the **positioning scheme.**
- The positioning scheme defines three modes: normal flow, floating, absolute
- **Normal flow:** The default one used, position the boxes consecutively **vertically (*block*)** or **horizontally *(inline*).**
  - *The inline elements are positioned consecutively: filling the width of the element they are contained in*
  - *The block elements one after another, with a line break*
- **Floating** (float): A box is placed in its normal position and then it is moved to the far right or left. **Other content flows around it**.
- **Absolute**: can be **displaced or fixed**. The box is out of the flow and is located at a certain position, but the other content does not flow, that is, the absolute box can overlap other boxes.

http://www.brainjar.com/css/positioning/default.asp

- Here you will find a detailed explanation of the positioning model

CSS – visual formatting model

Normal flow

- Block elements:
  - 100% width of parent element (which contains them)
  - Height depends on the content
- Inline elements**:**
  - Same line as long as there is space for them
  - Otherwise, moved to the next line
- https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Normal_Flow
  - Read the previous article in detail

CSS - Layout (layout)

The desired arrangement of elements on the page is one of the main objectives of the format using CSS

- However, it has traditionally been complex because it requires precise knowledge of the positioning algorithms used.
- The use of *float* to arrange the elements in columns, disrupts the normal flow of the other elements and makes them surround the floating element.
- Use *clear* to eliminate that behavior or put the *float* in a container element.

CSS 3 adds two new properties *flex* and *grid* that facilitate the creation of complex arrangements

- https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Introduction

***Frameworks* or additional libraries are used to simplify the design**

- Typically they make extensive use of javascript
- jQuery, bootstrap...

contents

XML

Extensible Markup Languaje (XML)

- **Metalanguage** to define particular markup languages: define and declare other markup languages.
  - It provides the syntax rules for creating languages that are **well formed and can be validated.**
  - That is to say, provide **rules to declare the DTD** of a particular markup language.
  - XML as a meta language provides a simplified SGML syntax
- Each particular language defined by XML **is an application of XML**. For example: XHTML, MathML, SVG.

XML instance and DTD

- Example: An XML language for notes

```
<?xml version="1.0"?>

<note>
        <to>Tove</to>
        <from>Jani</from>
        <subject>Reminder</subject>
        <message>no forget me Este weekend! </message>
</note>
```

- Associated DTD defined with the XML syntax

```
<!ELEMENT note (to, from, subject, message)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT message (#PCDATA)>
```

Data interchange

XML has been much used for data interchange

- Used by Web Services: interoperability or RPC systems implemented on top of the Web technologies
- Also by other applications

Nowadays JSON is the preferred choice:

- JSON: lightweight data-interchange format. **Text format**
- It is easy for humans to read and write.
- It is easy for machines to parse and generate
- Originally based on javascript. Now independent

contents

DOM

Document Object Model (DOM)

- **Neutral interface (API),** with respectot o platform and programming language, for the access and dynamical modification of content, structure and style of a structured language.
- What it is for? **Processing of structured languages** markup languages (HTML, XML, etc.).
- **Standard** to process these document
- When processing a document with DOM, the viewer (browser) usually allows **to dynamically display the modifications**.
- **3 specifications**: DOM Level 1, 2 y 3.
- A number of languages use it: **Javascript**, PHP, Java, etc
  - DOM is not only used by browsers

DOM Level 1

Functionality:

- Create documents.
- Navigate its structure.
- Add, remove o modify elements and attributes.

DOM documents have a tree structure.

Documents and elements are modeled after objecto (as in OOP): data types with properties and functions

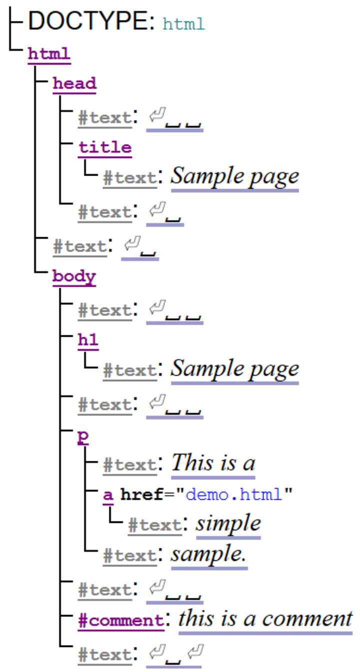
- Data are encapsulated in these objects.
- Objects provide function to modify the data and to interact with other objects

DOM tree and HTML

The browser represents an HTML document as a DOM tree in memory:

- **Attention**: Although used interchangably, **the DOM API and the DOM tree are two different concepts**. In principle, the **DOM API** provides functions to process structured documents. The "**DOM“ tree** is a data structure by which the document is represented internally by the browser. The DOM tree is constructed and processed using the DOM API functions.
- As the browser is analyzing (*parsing*) the document, the DOM tree is updated in RAM
- If a script modifies the DOM, the browser automatically applies the changes.

```
<! DOCTYPE html>
<Html>
 <Head>
  <Title> Sample page </ title>
 </ Head>
 <Body>
  <H1> Sample page </ h1>
  <P> This is a <a href= "Demo.html"> Simple sample
</a>. </ P>
  <! - this is a comment ->
 </ Body>
</ Html>
```



```
├DOCTYPE: html
└html
  ├head
  │ ├#text: ⏎␣␣
  │ ├title
  │ │ └#text: Sample page
  │ └#text: ⏎␣
  ├#text: ⏎␣
  └body
    ├#text: ⏎␣␣
    ├h1
    │ └#text: Sample page
    ├#text: ⏎␣␣
    ├p
    │ ├#text: This is a
    │ ├a href="demo.html"
    │ │ └#text: simple
    │ └#text: sample.
    ├#text: ⏎␣␣
    ├#comment: this is a comment
    └#text: ⏎␣⏎
```

DOM objects

DOM represents documents as a hierarchy of Node objects

- From this base class more specialized objects are derived: Element, Document, Text, etc.
- An Element represents a tag, therefore, its content will be the name of the tag.
- Text represents the text (which is usually between the start and end tags that contain it), therefore its content will be the text itself.

In addition, collections of objects are processed by the interfaces NodeList and NamedNodeMap

- NodeList is a doubly linked list of nodes.
- NamedNodeMap is a hash table of key-values.
- Any changes made in the document are automatically reflected in the NodeList and NamedNodeMap: if I get a list of nodes and then delete one of the nodes of the document, said node in the list will also be deleted.

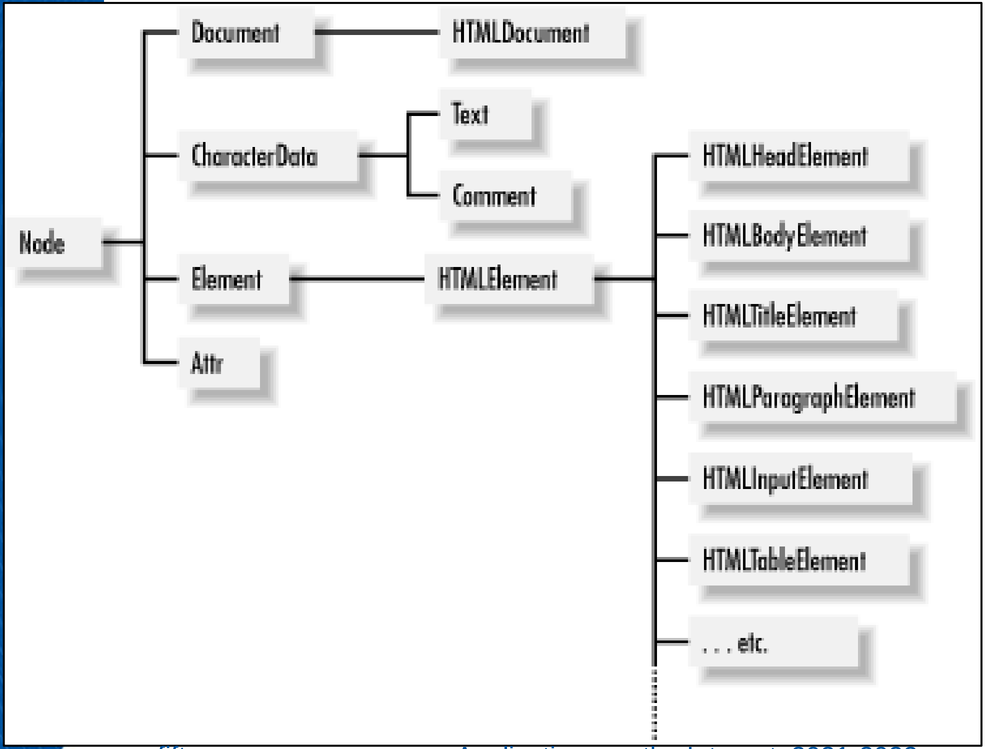
DOM hierarchies and extensions

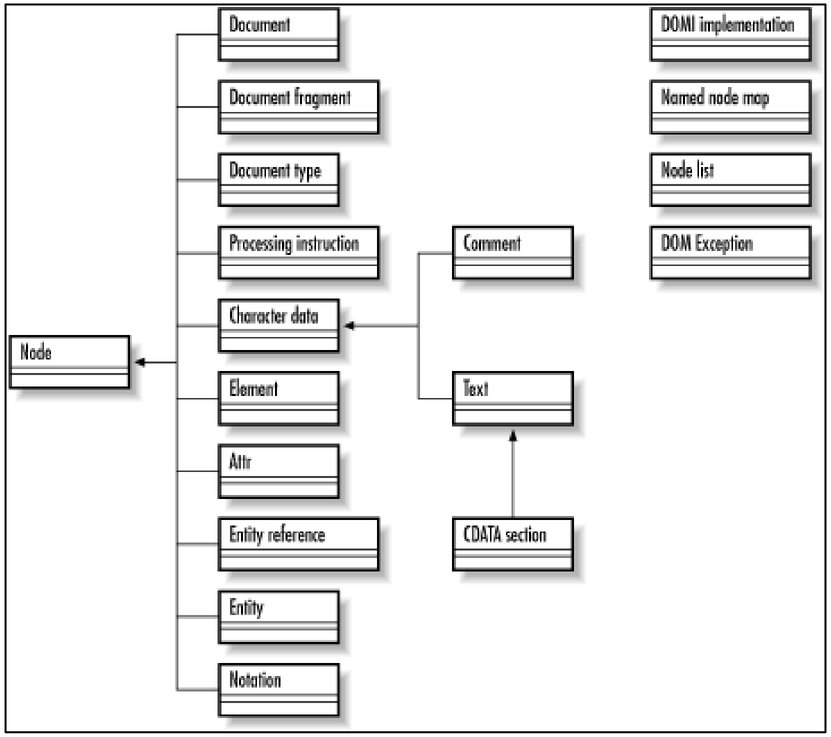
DOM is extended to support particular languages

- HTML DOM extends DOM to HTML
- Each class has its own interface (functions) that can be invoked programmatically

HTML DOM

SUN core





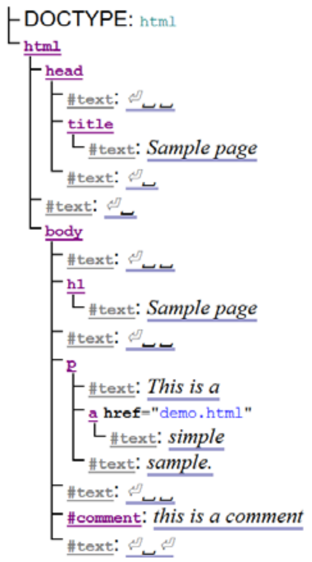
DOM-end comments

DOM is the standard interface for manipulating structured documents, including HTML.

- On the server side, there are implementations for major languages: PHP, Java, etc.
- On the client side, it is critical in the use of Javascript.
  - Still, problems arise due to browser compatibility.
  - One solution is to use additional libraries abstracting characteristics of DOM and ensuring that they work in different browsers, such as jQuery, Prototype or YUI.

Example DOM API javascript

```
<! DOCTYPE html>
<Html>
 <Head>
  <Title> Sample page </ title>
 </ Head>
 <Body>
  <H1> Sample page </ h1>
  <P> This is a <a href= "Demo.html"> Simple sample
</a>. </ P>
  <! - this is a comment ->
 </ Body>
</ Html>
```



variable *a* is a HTMLAnchorElement whose API is:
https: //developer.mozilla.org/en-US/docs/Web/API/HTMLAnchorElement

document.links returns a collection of nodes: NodeList access the first element as a vector

```
var a = document.links[0]; // Obtain the first link in
    thedocument
a.href= 'Sample.html'; // change the destination
    URL of the link
a.protocol= 'Https'; // just change the scheme part
    of the URL
a.setAttribute( 'href',' Http://example.com/ '); //
    change the content attribute Directly
```

Is this feature specific to HTMLAnchorElement or derived from another class?

Bibliography and extras

- HTML
  - 4.01 specifications: http://www.w3.org/TR/REC-html40/
  - https://developer.mozilla.org/en-US/docs/Web/HTML
- CSS
  - Specification: http://www.w3.org/TR/CSS2/
  - tutorials:

    http://www.brainjar.com/css/positioning/default.asp
    http://learn.shayhowe.com/html-css/getting-to-know-css/
    http://css-tricks.com/snippets/css/a-guide-to-flexbox/
  - Thousands of free templates: http://www.oswd.org/
    - There are many more sites offering free templates.
  - http://bootswatch.com
- DOM
  - Specification: http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html
  - https://developer.mozilla.org/en/docs/DOM
  - PHP: http://php.net/manual/en/book.dom.php