# Exam Preparation

## Web technologies basics

1. **Web Page** -
2. Document or information resource that is suitable for the World Wide Web.
3. Can be accessed through a web browser and displayed on a monitor or mobile device
4. This information is usually in HTML or XHTML format, and may provide navigation to other web pages via hypertext links.
5. Web pages frequently refer to other resources such as style sheets (CSS), scripts (JavaScript) and images into their final presentation.
6. Collection of related web pages containing web resources (web pages, images, videos, CSS files, JS files or other digital assets)
7. Common navigation between web pages
8. A website is hosted on at least one web server
9. Accessible via a network (such as the Internet)
10. All publicly accessible websites collectively constitute the World Wide Web.
11. **Web Application –**
12. Next level web sites
13. High interactivity
14. High accessibility (Cloud)
15. AJAX, Silverlight, Flash, Flex, etc.
16. Applications are usually broken into logical chunks called "tiers", where every tier is assigned a role
17. Desktop-like application in the web browser
18. Web applications on desktop (Windows 8)
19. **Web Browsers –**
20. Program designed to enable users to access, retrieve and view documents and other resources from the Web
21. Main responsibilities:

* Bring information resources to the user (issuing requests to the web server and handling any results generated by the request)
* Presenting web content (render HTML, CSS, JS)
* Capable of executing applications within the same context as the document on view (Flash)

1. **Layout Engines**
2. Software component that displays the formatted content on the screen combining:

* Marked up content (such as HTML, XML, image files, etc.)
* Formatting information (such as CSS, XSL, etc.)

1. It "paints" on the content area of a window, which is displayed on a monitor or a printer
2. Typically embedded in web browsers, e-mail clients, online help systems or other applications that require the displaying (and editing) of web content

* Trident-based

*Internet Explorer, Netscape, Maxthon, etc.*

* Gecko-based

*Firefox, Netscape, SeaMonkey, etc.*

* Blink-based

*Chrome, Opera*

* WebKit-based

*Safari, iOS, Maxthon, Chrome (up to v27), etc.*

* EdgeHTML (fork of Trident)

*Spartan (the new IE)*

*Windows 10 and Windows 10 (Mobile)*

1. **User Agent Strings –** Identify web browsers and their version. They usually can have some additional information like layout engine, user‘s operating system, etc.

****

* **Mozilla/5.0 –** a generic term which most modern browsers use (originally indicated Netscape)
* **Windows NT 6.3 –** Windows 8.1
* **WOW64 –** Windows-On-Windows 64-bit
* **AppleWebKit/537.36 –** Blink is a fork of WebKit

KHTML is the previous name of WebKit

* **Chrome/41.0.2272.118 –** real browser version
* **Safari/537.36 –** artifact against scripts sniffing

1. **Hardware servers**
2. **Physical computer** (a hardware system) dedicated to running one or more such services
3. **Servers** are placed in collocation centers

* Colocation facilities provide space, power, cooling, and physical security for the server

1. **The server may be**:

* Database server
* File server
* Mail server
* Print server
* VPS servers

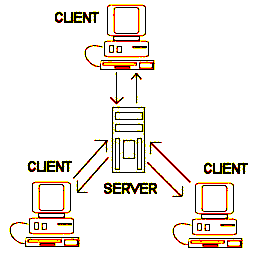
1. **Web Servers –** Apache, IIS, nginx, lightpd, etc.
2. All physical servers have hardware
3. The hardware is controlled by the operating system
4. Web servers are software products that use the operating system to handle web requests

* Web servers serve Web content

1. These requests are redirected to other software products (ASP.NET, PHP, etc.), depending on the web server settings
2. **Client-Server Architecture –** the client-server model consists of:
3. **Server –** a single machine or cluster of machines that provides web applications (or services) to multiple clients

* Examples:
* Web server running PHP scripts or ASP.NET pages
* IIS /Internet Information Services/ based Web server
* WCF based service
* Services in the cloud

1. **Clients –** software applications that provide UI - user interface (front-end) to access the services at the server.

* Examples:
* Web browsers
* WPF applications
* HTML5 applications
* Silverlight applications
* ASP.NET consuming services
* Web server (Apache, IIS) – Web browser
* FTP server (ftdp) – FTP client (FileZilla)
* Email service (qmail) – email client (Outlook)
* SQL Server – SQL Server Management Studio
* BitTorrent Tracker – Torrent client (uTorrent)
* DNS server (bind) – DNS client (resolver)
* DHCP service (wireless router firmware) – DHCP client (mobile phone/Android DHCP client/)
* SMB server (Windows) – SMB client (Windows)

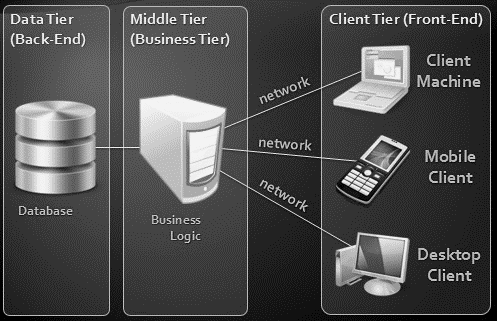
1. **3-Tier/ Multi-Tier Architectures –** it consists of the following tiers (layers):
2. **Front-end** (client layer)**:**

* Client software – provides the UI of the system

1. **Middle tier** (business layer)

* Server software – provides the core system logic
* Implements the business processes / services

1. **Back-end** (data layer)

* Manages the data of the system (database / cloud)

1. **Service-oriented architecture** - in the real world a "service" is:
   1. A piece of work performed by a service provider
   2. Provides the client (consumer) some desired result by some input parameters

* The requirements and the result are known
  1. Easy to use
  2. Always available
  3. Has quality characteristics (price, execution time, constraints, etc.)

1. **Cloud –** multiple hardware machines combine their computing power and resources

* Share them between multiple applications
* To save costs and use resources more efficiently
  1. **Public clouds**
* Provide computing resources on demand
* Publicly in Internet
* Paid or free of charge (to some limit)
* Amazon AWS, Google App Engine, Microsoft Azure, Rackspace, PHPFog, Heroku, AppHarbor
  1. **Cloud Computing Models**
* **Infrastructure as a Service (IaaS)**
* Virtual machines in the cloud on demand
* Users install the OS and software they need
* **Platform as a Service (PaaS)**
* Platform, services and APIs for developers
* E.g. Java + JBoss + JSF + JPA + MongoDB or JavaScript + Node.js + MongoDB + RabbitMQ
* **Software as a Service (SaaS)**
* Hosted application on demand (e.g. WordPress)

## Tools for HTML, CSS and JS

1. **Coding Tools –** there are many, many coding tools for HTML, CSS and JavaScript
   * + 1. They are cross-platform
       2. Some of the most used are:

* Sublime Text 2/3 – a text editor with a little ready-to-use functionality. Contains free plugins, it’s easily customizable
* Atom.io – a hackable text editor for the 21st Century. Build with HTML, CSS and JavaScript for development of HTML, CSS and JavaScript, open-source
* Jetbrains WebStorm – ready-to-use IDE for JavaScript development, including web development with HTML and CSS. It’s a paid software.
* Microsoft Visual Studio – ready-to-use IDE for web development. It has a free community version, yet it is paid.
* Eclipse, Aptana, Komodo IDE, Notepad++

1. **Browsers and Browser Tools**
2. **Browsers**

* Internet Explorer
* Google Chrome
* Mozilla Firefox
* Apple Safari
* Opera

1. **Browser Tools**

* Firebug
* WebInspector
* WebDeveloperToolbar
* F12 (IE)
* Inspect
* Console
* Bookmarklets

1. **Drawing and Slicing Tools**

* Adobe Photoshop
* Adobe Fireworks
* Gimp
* Paint.NET

## HTML Fundamentals

1. **Hypertext Markup Language –**
2. **HTML - H**yper **T**ext **M**arkup **L**anguage

* A notation for describing
* **Document structure** (semantic markup)
* **Formatting** (presentation markup)
* Looks like:
* A Microsoft Word document

1. The markup tags provide information about the page content structure
2. An HTML document consists of many tags
3. HTML document must have an **.htm** or **.html** file extension
4. HTML files can be created with text editors:

* Notepad, Notepad++, Sublime Text

1. Or HTML editors (WYSIWYG Editors):

* Microsoft WebMatrix
* Microsoft Expression Web
* Microsoft Visual Studio
* Adobe Dreamweaver

1. **HTML – Past, present, future**
2. 1991 – HTML first mentioned – Tim Berners-Lee – HTML tags
3. 1993 – HTML (first public version, published at IETF)
4. 1993 – HTML 2 draft
5. 1995 – HTML 2 – W3C
6. 1995 – HTML 3 draft
7. 1997 – HTML 3.2 – “Wilbur”
8. 1997 – HTML 4 – ”Cougar” – CSS
9. 1999 – HTML 4.01 (final)
10. 2000 – XHTML draft
11. 2001 – XHTML (final)
12. 2008 – HTML5 / XHTML5 draft
13. 2011 – feature complete HTML5
14. **HTML Terminology**
15. Concepts in HTML

* Tags
* Opening tag (mark the start of an HTML element) and closing tag (mark the end of an HTML element)
* The smallest piece in HTML
* Start with “<” and end with “>”
* Attributes
* Properties of the tag
* Size, color, border, etc…
* Put directly in tags
* Has value surrounded by “ “ or ‘ ‘ – the value is always a string.
* Some of the attributes are common for every HTML element (*id, class, name, style)*
* Some are specific (*src* of the *img* element)
* Elements
* Combination of opening, closing tag and attributes

1. **HTML Document Structure**

Some elements are essential to each HTML Document:

* *html, head, body, doctype*

1. The *html* element

* Used to mark the beginning and ending of a HTML document
* All the content of the web page is inside this tag: <html> … </html>

1. The *head* element – contains markup that is not visible to the user, but it helps the browser to render correctly the HTML document. It contains:

* Styles, scripts
* Declare encodings
* The *title* tag – the text in the tab of a browser.

1. The *body* element and *Doctype*

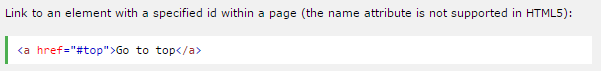
* *Body* element contains all the visible to the user markup:
* Headings, text, hyperlinks, images, etc…
* Textboxes, sliders, buttons…
* *Doctype* is kind of the validator of the page – tells the browser in which version of HTML the page is written.
* **HTML5 Doctype =** <!DOCTYPE html>

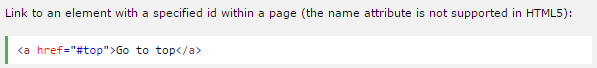
1. **HTML Common Elements –**
2. Text formatting tags modify the text between the opening tag and the closing tag:

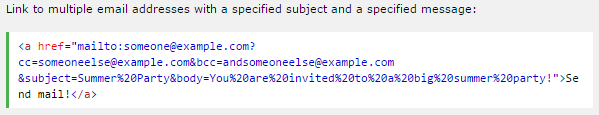
|  |  |
| --- | --- |
| **<b></b>** | **bold** |
| **<i></i>** | ***italicized*** |
| **<u></u>** | **underlined** |
| **<sup></sup>** | **Samplesuperscript** |
| **<sub></sub>** | **Samplesubscript** |
| **<strong></strong>** | **strong** |
| **<em></em>** | ***emphasized*** |
| **<pre></pre>** | **Preformatted text** |

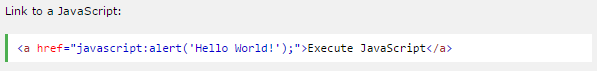
1. Hyperlink Tags







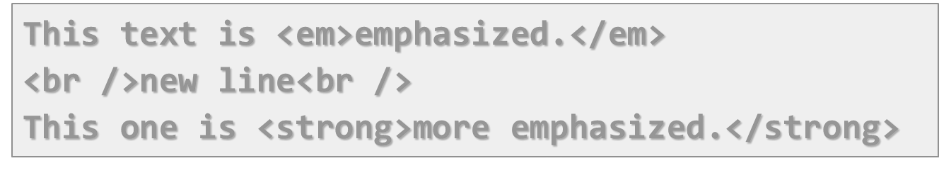




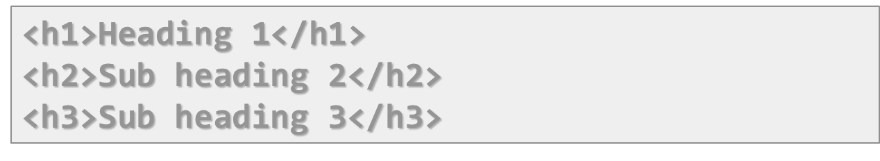
1. Image Tags



1. Text formatting tags



1. Heading Tags (h1 – h6)



1. Paragraph Tags



1. Sections: div and span

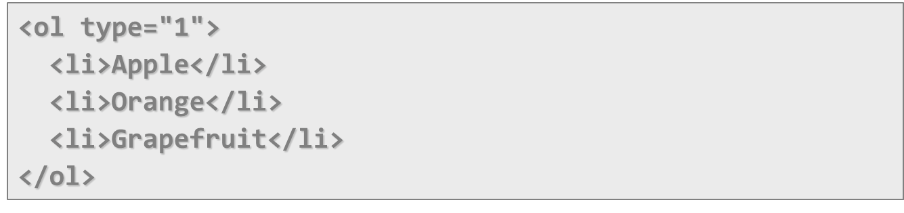


* <div> creates logical divisions within a page
* Block element
* Used with CSS
* <div> is used with IDs for styling: <div id=”header”> … </div>
* Example:



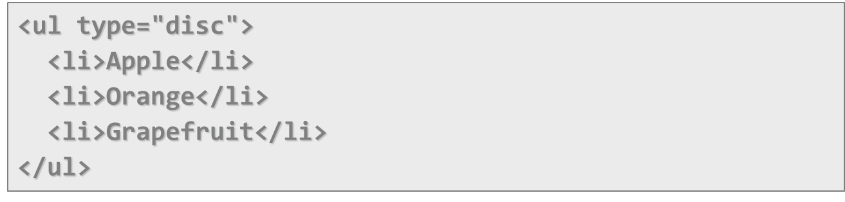
* <span> inline style element – useful for modifying a specific portion of text. Mainly used to style parts of a text



1. Create an Ordered List using <ol></ol>:

Attribute values for type are 1, A, a, I, or i

1. Create an Unordered List us <ul></ul>



Attribute values for type are: disc, circle or square

1. Create definition list using <dl>

The <dl> tag is used in conjunction with [<dt>](http://www.w3schools.com/tags/tag_dt.asp) (defines terms/names) and [<dd>](http://www.w3schools.com/tags/tag_dd.asp) (describes each term/name).



## HTML Tables

Tables represent tabular data. A table consists of one or several rows. Each row has one or more columns.

Tables are comprised of several core tags:

**<table></table>:** begin/end table definition

**<tr></tr>:** create a table row

<**td></td>:** create tabular data (cell)

Tables should **not** be used for layout – use CSS floats and positioning styles instead.

|  |  |
| --- | --- |
| 1. **Data cells and Header cells** 2. Two kinds of cells in HTML tables  * **Data** cells – containing the table data * **Header** cells – used for the column names or some more important cells.  1. Why two kinds of cells?   Used to **semantically** separate the cells |  |
| 1. **Complete HTML Tables**   Table rows split into three semantic sections: **header, body** and **footer**   1. **<thead>** denotes table header and contains <**th>** elements, instead of <**td>** elements 2. **<tbody>** denotes collection of table rows that contain the very data 3. **<tfoot>** denotes table footer but comes BEFORE the <tbody> tag 4. **<colgroup>** and <**col>** define columns (used to set column widths) 5. **<caption>** tag defines a table caption.   The <caption> tag must be inserted immediately after the <table> tag.  **Note:** You can specify only one caption per table. |  |
| 1. **Nested Tables**   Table “cells” (**<td>)** can contain **nested** tables (tables within tables): |  |
| 1. **Complex Tables /with Padding, Spacing, etc./**   Tables have two attributes related to space   1. **Cellspacing –** defines the empty space between cells.      1. **Cellpadding –** defines the empty space around the cell content |  |
| 1. **Column and Row Span**   Cells have two attributes related to merging   1. **Colspan –** defines how many columns the cell occupies      1. **Rowspan –** defines how many rows the cell occupies |  |

## HTML Forms and Frames

1. The **primary** method for gathering data from site visitors.

HTML Forms can contain:

1. **Text** fields for the user to type
2. **Buttons** for interactions like “**Register**”, “**Login**”, “**Search**”
3. **Menus, Sliders,** etc…
4. **How to create a HTML form?**
5. Create a form block with:

<**form></form>**

**Example:**

<form name="myForm" method="post" action="path/to/some-script.php">

...

</form>

* The “method” attribute tells how the form data should be sent – via GET or POST request
* The “action” attribute tells where the form data should be sent

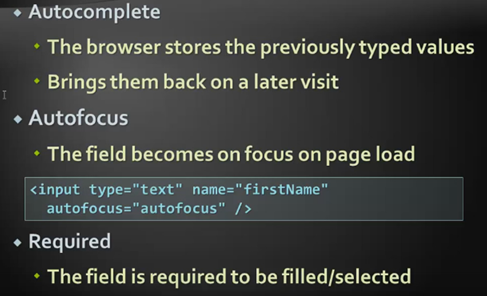
|  |  |
| --- | --- |
| **<form name="...">** - the name of the form | The name attribute specifies the name of an <input> element.  The name attribute is used to reference elements in a JavaScript, or to reference form data after a form is submitted.  Note: Only form elements with a name attribute will have their values passed when submitting a form. |
| **<form method="...">** - how to send data (post) | The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute).  The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post"). |
| **<form action="...">** - where to send data (to urls) | The action attribute specifies where to send the form-data when a form is submitted. |
| **<form value="...">** - | The value attribute specifies the value of an <input> element. The value attribute is used differently for different input types:   * For "button", "reset", and "submit" - it defines the text on the button * For "text", "password", and "hidden" - it defines the initial (default) value of the input field * For "checkbox", "radio", "image" - it defines the value associated with the input (this is also the value that is sent on submit)   (In HTML5, the action attribute is no longer required.) |

|  |  |
| --- | --- |
| **<input type="text" />** - textbox |  |
| **<input type="reset" />** - reset button for the form |  |
| **<textarea>** - multiline textbox  + атрибут placeholder (изписва текст, който изчесва, когато тръгнем да въвеждаме) |  |
| **<input type="password" />** - \*-textbox |  |
| **<input type="submit" />** - submit button |  |
| **<input type="image" />**- gets coordinates from image |  |
| **<input type="button" />** - with javasctipt |  |
| **<input type="checkbox" />**- checkbox    value – информацията, която се изпраща на сървъра ако чекнем  (fruit==apple) |  |
| **<input type="radio" name="..." /> - radio buttons group by "name"** |  |
| **<select><option value="..." selected="selected"></option></selected>** - dropdown menu |  |
| **<select multiple="multiple"></select>** - multiple choice menu |  |
| **<input type="hidden" />**- hidden data not seen by user  Потребителя не попълва и не вижда, но се изпращат до сървъра |  |
| **<label for="example">**- are used to associate an explanatory text to a form field using the fields ID  тук липсва name  **name** и **value** се изпращат на сървъра | * Clicking on a label **focuses** its associated field * Checkboxes are **toggled** * Radio buttons are **checked** * Labels are * Both a **usability** and **accessibility** feature * Required in to pass accessibility validation |
| **<fieldset>** - are used to enclose a group of related form fields:  **<legend** **>** - caption for the fieldset | C:\Users\M.Boykov\Desktop\2016-04-14_155459.jpg  Разделя тематично различните форми за попълване: Лични данни, допълнителни въпроси,…др. |

1. **Sliders and Spinboxes**

|  |  |  |
| --- | --- | --- |
| **<input type="range">**- spinbox (does not work in Firefox)  **<input type="number">** - slider (does not work in Firefox) |  |  |

1. **Field Attributes from HTML5**



1. **Input Fields with Validation**
2. **Email –** provides a simple validation for email

* Can be passed a pattern for validation
* In a mobile device brings the email keyboard

<input type="email" required="true"   
pattern="[^ @]\*@[^ @].[^ @]"/>

1. **URL –** has validation for url

In a mobile device brings the url keyboard

<input type=”url” required=”true” />

1. **Telephone**

Brings the numeric keyboard

<input type=”tel” required=”true” />

1. **Tab Index**
2. The **tabindex** HTML attribute controls the order in which form fields and hyperlinks are focuses when repeatedly pressing the **TAB** key.

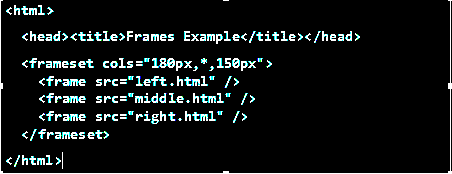
* **tabindex=”0”** (zero) – “natural” order
* If **X<Y,** then elements with **tabindex=”X”** are iterated before elements with  **tabindex=”Y”**
* Elements with negative **tabindex** are skipped, however, this is not defined in the standard.

<input type="text" name="second" tabindex="10" />

<input type="text" name="first" tabindex="5" />

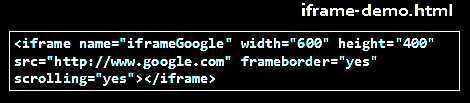
1. **HTML Frames -** <frameset>, <frame> and <iframe>
2. **Frames** provide a way to show multiple HTML documents in a single Web page.

* The page can be split into separate views (frames) horizontally and vertically
* Frames were popular in the *early* ages of HTML development, but now their usage is rejected
* Frames are not supported by all user agents (browsers, search engines, etc.)

A <**noframe>** element is used to provide content to non-compatible agents.

Note the **target** attribute applied to the **<a>** elements in the left frame.

1. **Inline frames** provide a way to show one website inside another website:



## Semantic Web

How to Use HTML Elements properly?

1. **The Elements of a Web page**

A Web Page consists of:

* HTML markup
* CSS rules
* **JavaScript code**
* **JS libraries**
* Images
* Other resources
* Fonts, audio, video, Flash, Silverlight, etc…

1. The HTML is used to define the **content** of a Web page

! Not the layout

! Not the decorations

1. HTML’s role is to present the information in a **meaningful** manner

* Like a paper document
* Define headers, paragraphs, textboxes, etc…
* Not define size, color and/or positioning

1. **Cascading Style Sheets (CSS)** is a way to make a Web page look pretty

* Define **styling rules**
* Fonts, colors, positioning, etc.
* Define the layout of the elements
* Define the presentation

The CSS files are attached to a web page and the browser applies these styles to elements.

1. **JavaScript** is the programming language for the Web

* Makes the Web pages dynamic
* Dynamically adding / removing HTML elements, applying styles, etc.
* Modern JavaScript UI libraries provide UI components like dialog boxes, grids, tabs, etc.

Like CSS, the JavaScript files are attached to a web page

There are also other resources that are needed for a Web page to run properly: Images, fonts (glyph icons), audio, video files

Flash / Silverlight / ActiveX objects

**The Semantic HTML**

1. **Semantic HTML is:**
2. The use of HTML markup to reinforce the semantics of the information in Web page

* Make the content understandable for computers
* Rather than merely to define its presentation
* A kind of **metadata** about the HTML content

Semantic HTML is processed by regular Web browsers and other user agents

* CSS is used to suggest its presentation to human users

1. Easier to read by developers, parsers, bots, machines, AIs
2. A way to show the search engines the correct content
3. **How to Write Semantic HTML?**

Just follow some guidelines when creating a Web site

1. Use HTML5 semantic tags

* **<header>, <nav>, <section>, <article>, <aside>, <footer>**

1. Use Headings when you need to structure the content into sub-headings

* In increasing order, starting with <h1>
* Do not use empty tags > Like a clearing **<div>**

1. **HTML5 Semantic Tags**

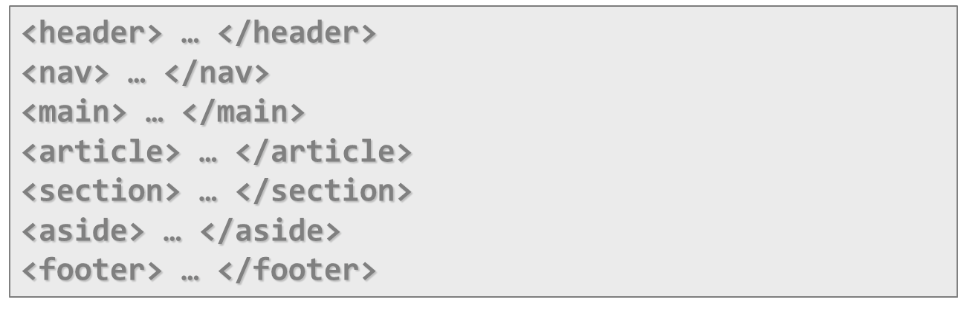
HTML 5 introduces **semantic structure tags**

****

This is a common Web page structure, used in **90%** of the web sites

This can be created using all kind of HTML elements - <div>, <span>, even <p>. Browsers will render invalid / wrong / pseudo valid HTML

1. **The correct way: use the HTML5 semantic tags:**



1. **<main>**

Specifies the main content of a document.

There must not be more than one <main> element in a document

1. <**header>**

Site header or section header or article header

Could include navigation (<nav>)

1. <**footer>**

Site footer (sometimes can be a section footer)

Providing author, copyright data, etc.

1. <**nav>**

Defines a set of navigation links

E.g. site navigation (usually in the header)

1. **<aside>**

Content slightly related to primary content

E.g. sidebar (usually on the left or on the right)

1. **<section>**

Grouping of content usually with a heading, similar to chapters

Site sections (e.g. news, comments, links, …)

1. **<article>**

Independent content such as blog post or an article (e.g. news item)

1. **<details> + <summary>**

Specifies additional details that the user can view or hide on demand (accordion-like widget)

1. **<time>**

Specifies date / time (for a post / article / news)

1. **<mark>**

Defines marked/highlighted text

1. **<figure>**

Grouping stand-alone content (video or image)

Figure (a figure, e.g. inside an article)

1. **<figcaption>**

A caption of a figure (inside the **<figure>** tag)

1. **<video>**

Video element (uses the built-in player)

1. **<audio>**

A standard for playing audio files (built-in player)

1. **<dialog>**

Defines a dialog box or window

1. **<meter> / <progress>**

Defines a scalar measurement within a known range (a gauge) or task progress

1. **<output>**

Defines the result of a calculation

1. **<wbr>**

Defines a possible line-break

1. **Other Semantics** (Headings, ems, strongs)
2. Headings

Always use headings (<h1> - <h6>) when you need a heading or title

Google uses it to mark important content

1. Strong <**strong>** vs Bold <**b>**

<**b>** does not mean anything. It just makes the text bolder

<**strong>** marks the text is “*stronger”* than the other, surrounding text

1. Emphasis <**em>** vs Italic <**i**>

Emphasis does not always mean, that the code should be *italic.* It could be bolder, italic and underlined.

The styles for the emphasis text should be set with CSS, not by HTML.

1. **Accessibility**
2. Craft content minding disabled users

* Blind – include text equivalents of images, use labels in forms
* Colorblind – do not convey information using color only
* Visually impaired – avoid small font sizes
* Epileptic – avoid flashing content (3Hz or more)
* Physical disabilities – avoid functionality that relies only on the mouse or keyboard

1. Why implement accessibility?

* Some accessibility features are mandatory for government sites in some countries (US, NL, SW)
* “Everyone gets visited by a very important blind user, named Google”
* Some SEO and accessibility considerations overlap

1. **Search Engine Optimization –** Getting ahead in search engines

Search engines use so-called “crawlers” to get the content of the page and index it.

The crawlers weigh the data on the page:

1. <**title>, page URL** and **headings** have great weight
2. Links from highly valued pages to your page increase its value
3. Add alt text to images
4. Use relevant keywords in the content and <**meta>** tags

No SEO technique will replace good content.

1. **Structured Data Markup –** Annotate your content so machines can understand it

A standard way to annotate your content so machines can understand it

Google (and other search engines) can:

1. Use that data to index your content better
2. Present it more prominently in search results
3. Provide answers from the Knowledge Graph
4. **Three alternative formats:**

* **Microdata and RDFa**
* Define new HTML attributes
* JSON-LD
* Newest and simplest markup format
* Embed a block of JSON data inside a script tag

## Others we should now

1. The **Document Object Model** (**DOM**) is a [cross-platform](http://en.wikipedia.org/wiki/Cross-platform) and [language](http://en.wikipedia.org/wiki/Programming_language)-independent *convention* for representing and interacting with [objects](http://en.wikipedia.org/wiki/Object_(computer_science)) in [HTML](http://en.wikipedia.org/wiki/HTML),[XHTML](http://en.wikipedia.org/wiki/XHTML), and [XML](http://en.wikipedia.org/wiki/XML" \o "XML)documents.
2. **Extensible Markup Language (XML)** is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable.
3. **Extensible Hypertext Markup Language (XHTML)** is a family of XML markup languages that mirror or extend versions of the widely used Hypertext Markup Language (HTML), the language in which Web pages are formulated.
4. The **Domain Name System** (**DNS**) is a [hierarchical](http://en.wikipedia.org/wiki/Hierarchical) distributed naming system for computers, services, or any resource connected to the [Internet](http://en.wikipedia.org/wiki/Internet) or a [private network](http://en.wikipedia.org/wiki/Private_network). It associates various information with [domain names](http://en.wikipedia.org/wiki/Domain_name) assigned to each of the participating entities.