The structure of HTML documents

An HTML document is composed of the following parts:

1. **Document Type declaration**
2. **Root element**
3. **Descriptive head part**
4. **The content body part**

The root element must contain all document content except for the document type declaration. The elements must be opened and closed accordingly (see on the right). These document parts may occur only once in a particular HTML document.

```
<!DOCTYPE html>
<html>
<head>
        descriptive part
</head>
<body>
        content part
</body>
</html>
```

Declaring the document as HTML version 5 and later

<!DOCTYPE html>

Doctype is a remnant of SGML document type definitions defined in SGML metalanguage, which was used with earlier HTML version. However, this is important to declare for the web browser to know which HTML version the document is using and to ensure correct parsing (interpretation) of the document!

Syntax -> uppercase letters for DOCTYPE and no space between it and the preceding exclamation mark (!). After a space, the abbreviation “html” (means HTML 5 and later versions) is written in small letters.

This is a much-appreciated simplified definition; see http://www.w3.org/QA/2002/04/valid-dtd-list.html for W3C valid doctypes for previous versions of HTML.

Root element

<html>…</html>

The **root element** of **HTML** and XHTML documents is `<html>…</html>`. When parsing a document, web browsers assume that it has a root element. All other elements of the document must be child/descendant elements in this root element. Child (descendant) means that an element is within the opening and closing tags of another element (the descendant can be at any hierarchical level), the parent (ancestor).

You are encouraged to use a *lang* attribute in the root element to define the main language of your document’s content. It is not obligatory and has little practical consequences at present other than semantical meaning. Browsers may use this information to aid

pronunciation in text readers or help translation utilities. Here, defining Finnish as the language of the document content.

```
<html lang="fi">…</html>
```

Other language codes: en = English (fi = Finnish , sv = Swedish, de = German, fr = French, es = Spanish … etc.)
http://reference.sitepoint.com/html/lang-codes

You can also use the lang attribute within the content part of the document for any specific element containing text, e.g.:

```
<p lang="en">This paragraph is in English.</p>
```

This could be a useful declaration of the language of a block of text, if the document otherwise is using a language other than English.

Head

<head></head>

Head element as the first element in the root element, should contain declarative/ descriptive information about the document, a collection of metadata. `<head>…</head>`

The head part **must not** contain elements that are meant for displaying as content in the browser window! There is only one head element in a document.

The head should always include a page **title** with the `<title>`*insert title text*`</title>` element. The head should also have a declaration of the document's character encoding. UTF-8 encoding is a recommendation for HTML 5 documents and can be declared simply:
`<meta charset="UTF-8">`

Body

<body></body>

Body element represents the actual content of a document that is to be displayed in the browser window. `<body>…</body>`

Some content in the body, in specific elements, may not be displayed, e.g. comments, scripts…

The body is the second element in the root element and there is only one body element in a document.

Basic HTML 5 document structure:

```
<!DOCTYPE html>

<html lang=”en”>

  <head>
    <meta charset="UTF-8">
    <title>Example 01</title>
  </head>

  <body>
    <h1>Main heading for the page</h1>
    <p>This is my first document.</p>
  </body>

</html>
```

http://www.w3.org/wiki/HTML/Training/HTML_Document

Elements in the Head

The HEAD part contains information about the document; this content is not displayed directly in the browser window. *<head></head>*

The head should not contain text outside its child elements or text that are meant for display as document content in the viewport!

Title

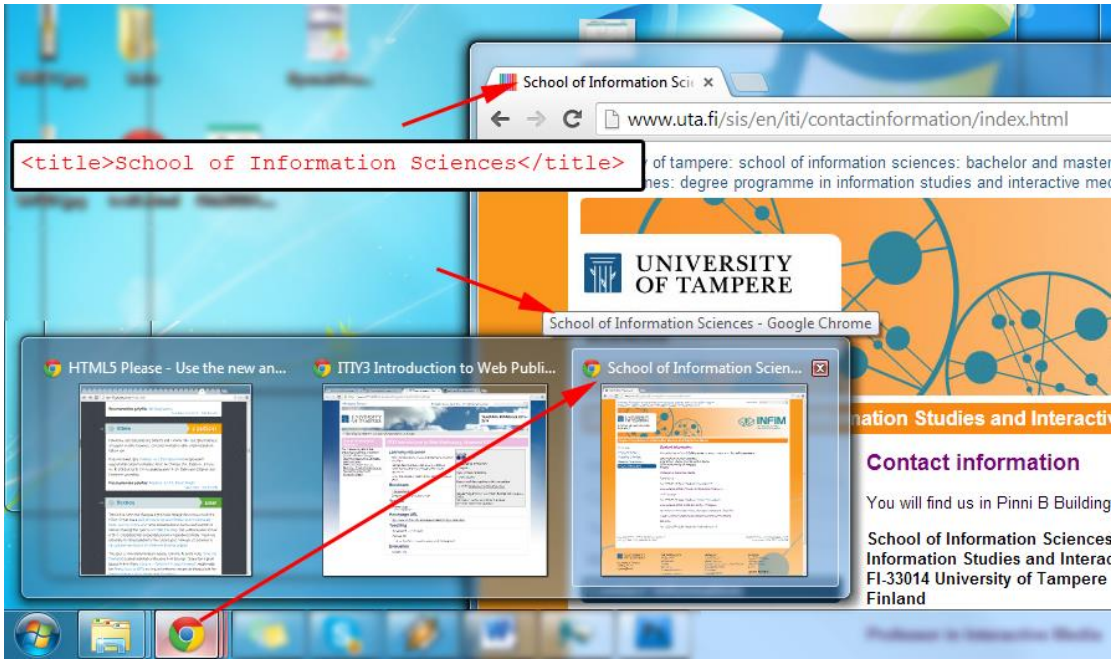
Every HTML document should have a **TITLE** element (only one) to identify the document and/or give an idea about the contents of the document if used in a different context: e.g.

<title>*Simple HTML structure****</title>.***

The title is typically displayed in the top bar of the browser, nowadays in the browser tab (and in the taskbar) outside the page content. The title should not be too long, preferably shorter than 72 characters! They should contain only ASCII characters and special characters should be referred to by their character entities… -> **&** (ampersand) = ***&*** or **@** = ***@*** see more below.

The title may be used for bookmarking in user agents. Search engines partially rely on them also. Search result listings may display page titles as they are besides using document headings and other selected content.

The title should be compact and descriptive of the organisation or topic that the document represents. A good title makes the page easier to find!



Meta

You may need to include some extra metadata information in the document for web browsers, web services, and server scripts. Typically, the **meta element** is used to define such information.

http://www.w3.org/TR/html5/document-metadata.html#the-meta-element

The document can contain multiple meta elements that provide different information. Each meta element should have a separate meaning defined by various attributes. Meta elements cannot be combined together.

One of the important meta elements is the definition of **character encoding** that is used in the document. **HTML 5 encourages to use UTF-8 unicode character encoding** in HTML 5 documents**:**

```
<meta charset="UTF-8">
```

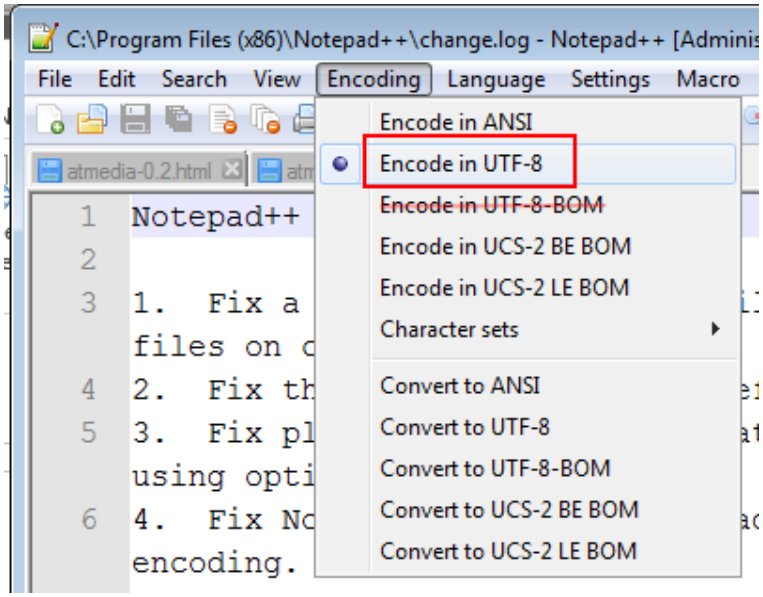
http://en.wikipedia.org/wiki/Character_encodings_in_HTML

Modern web browsers determine automatically which encoding the document is using. In some web browsers, a default encoding can be also set manually for documents without a meta element for character encoding. Google Chrome has a new algorithm, which is mostly right in detecting encoding. However, it is always safer to declare the encoding in the head part of each document to avoid misdetection. Characters in the document may not display correctly if the wrong encoding is determined or detected for a document.
https://www.w3.org/TR/html5/syntax.html#encoding-sniffing-algorithm

If you use UTF-8, you need to declare document charset as UTF-8, and make sure you select the proper encoding (UTF-8) for saving the file too.

Notepad++ has a separate menu for defining the *Encoding* of the file. For HTML5 documents **UTF-8** (byte-order mark, BOM is NOT needed!) should be selected as the encoding.

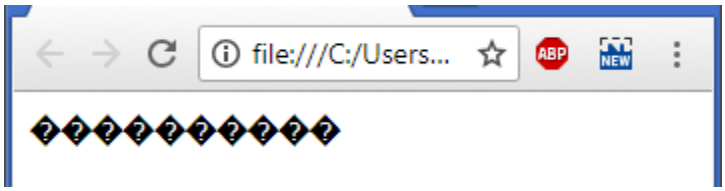
(image below Notepad++)



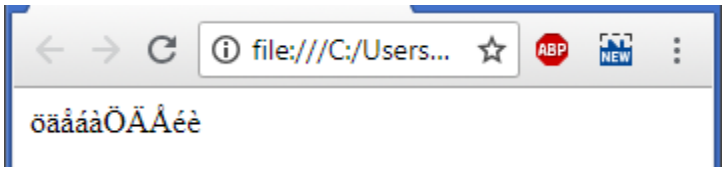
http://www.w3.org/wiki/HTML/Training/Metadata

If you have the wrong encoding for your document, use the conversion commands to convert your document.

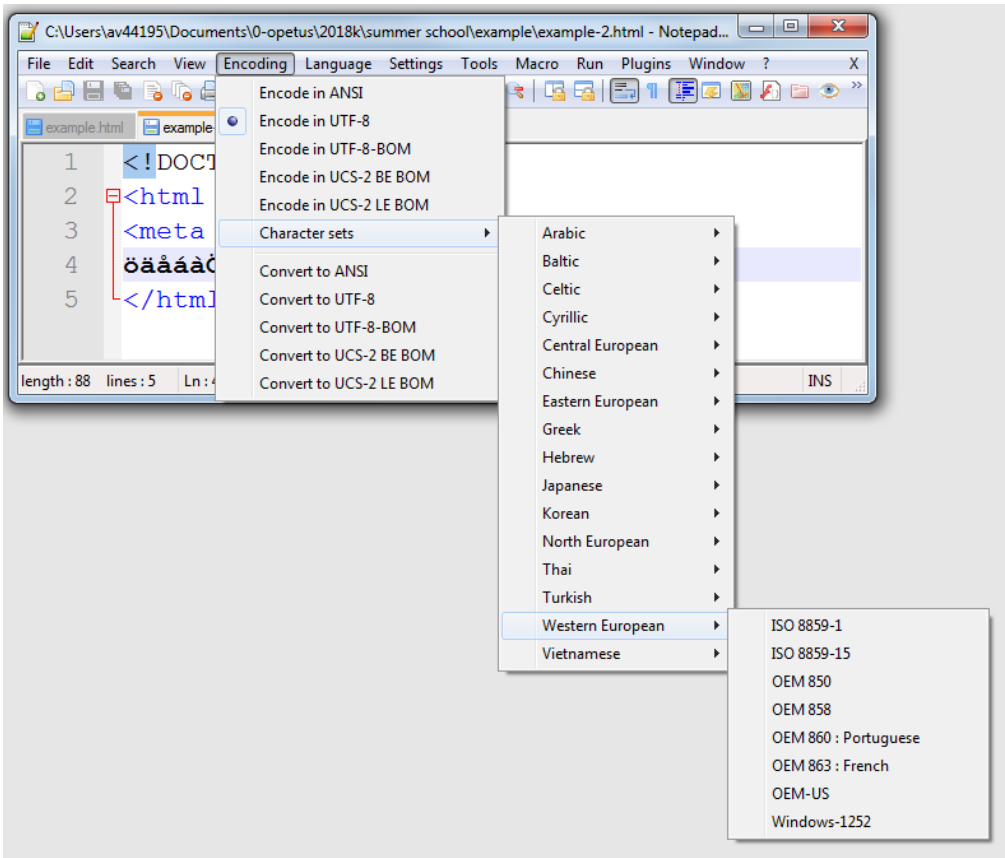
non-ASCII characters and ASCII encoded document with UTF8 charset declaration <meta charset="UTF-8"> set in the head:



non-ASCII characters in a UTF-8 encoded document with <meta charset="UTF-8"> set in the head:



With certain specific languages, you may need to use another encoding in your document than UTF-8, see link below (item 9 lists other suggested encodings). You may need to convert and save files with language specific character sets (see below in the image, Encoding menu Notepad++ > Character sets > choose the language or area specific option).



Declare the chosen encoding in the meta element! e.g. <meta charset="ISO-8859-1">, see point 8 at https://www.w3.org/TR/html5/syntax.html#encoding-sniffing-algorithm

Keep a copy of your original text in case the encoding operation goes wrong and the changes cannot be undone!

Character Entity References

Besides defining proper encoding in the document, sometimes it is useful and in some cases needed that you define certain characters by using character reference codes. Any alphanumeric characters may be expressed with character references in HTML documents. The syntax for character codes is **&**entity**;** where "entity" is the name or code of the character. The character entity code, when recognized by the web browser, appears as the corresponding character in the HTML document and not as "code".

Some characters are reserved for HTML markup and you should not use them in the content.

Reserved characters for HTML markup

&amp;	&	ampersand (strictly reserved for markup)
&lt;	<	less-than sign (strictly reserved for markup)
&gt;	>	greater-than sign (reserved for markup)
&quote; or "	"	straight quote (reserved for markup)
' or '	'	apostrophe (reserved for markup)

other examples of character codes

		character space (non-breaking space)
&#64;	@	commercial "at" sign
&copy;	©	copyright (also ©)
etc…		

Replace a character with its character reference in the HTML document when you need to add characters that are not on your keyboard, characters used strictly by HTML. You should avoid using the strictly reserved characters (& and <) outside the markup in the text content. Character entities cannot be used for creating HTML element markup. The characters <ul> will show up as plain text <ul> in the content and will not be interpreted as HTML markup by the web browser.

More on character codes:

http://dev.w3.org/html5/html-author/charref
http://www.digitalmediaminute.com/reference/entity/index.php
http://www.cookwood.com/html/extras/entities.html

Meta description

<meta name="**description**" content="This page is about HTML. Its syntax and example usage of various elements">

Additional text descriptions added by this meta element, may be used by some web search tools when indexing a document. The description should describe the content of the document briefly and accurately. This is something that is not visible as content and users may only see it if they look at the source code of the page.

There are also **other** meta elements that may be used to add e.g. a list of keywords, info about the author, etc.

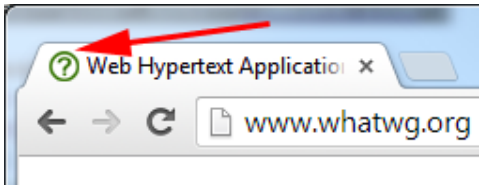
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta

Other elements in the HEAD

- *<styles></style>* style element for document specific **Style Sheet** definitions, which may be used to control document layout (CSS).
- *<script></script>* for **JavaScripts** code embedded within the element to perform actions (user interactivity, dynamic content)… Scripts may be added to the document within the script tags *<script> script is here </script>*. Alternatively external script files may be linked with the src attribute *<script src="somescript.js"></script>.* The script element may be used in both head and body parts of the document.
- *<link>* link-element that can link or append different resources to the document, such as an external style sheet: <link rel="stylesheet" href="mystylesheet.css"> (where mystylesheet.css is an external style file).

The link-element may be used to add an **icon** (image) to the page, displayed next the address bar or in the tab of the browser showing the page, and in bookmarks, or added to the home screen of mobile devices. This icon is a miniature logo, graphical id of the page. Site icons help identifying your site and its pages.

The icon may be an image file in the size of 16x16, 32x32 (pixels) (also other sizes), with .ico or .png image file format endings.



<link rel="icon" href="/images/icon.ico">

http://blog.whatwg.org/the-road-to-html-5-link-relations#rel-icon
https://html.spec.whatwg.org/#rel-icon:rel-icon
Make your own icons from your image: http://realfavicongenerator.net/

More on elements of the HEAD
http://www.whatwg.org/specs/web-apps/current-work/multipage/semantics.html#the-meta-element
http://www.w3.org/wiki/HTML/Training/Metadata

Elements in the BODY element

`<body></body>`

contains the actual contents of the page…

All the elements in the normal **document flow** (visible elements and their order in the document) are inside the body. Remember though, that not necessarily all elements in the body are visible in the browser window (viewport) (e.g. scripts, elements hidden with CSS, or with comments).

Text Elements

All text that is **not** enclosed with the <...> brackets are interpreted as plain text and showed as part of the visible document content. Text elements can organize text into visual **blocks** (block-level) forming distinct blocks of text within the page. Other elements reside **inline** or at **text-level.** These elements are used within text blocks, to mark smaller chunks of text, to mark phrases.

Typical block-level elements are for text entities forming blocks of text, with a default presentation starting text in the element with a **new line**. These are the main building blocks of documents. Many of these elements has a specific default appearance / formatting (applied by the web browser) which can be altered with CSS style rules.

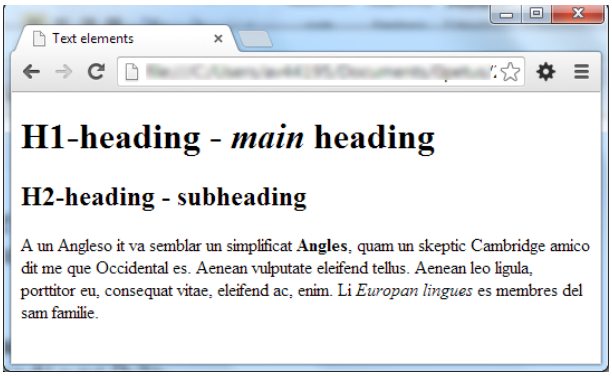
HTML 5 present alternate semantic categories for elements based on the type of content they are used for -> https://w3c.github.io/html/dom.html#kinds-of-content . Note that for practical reasons and for simplicity, we will use a basic block and text-level division model of elements (there are also other types), which are based on presentation models rather than semantics. The meaning of elements will not be ignored, however!

HTML elements have **semantic meanings,** which should determine the context how and where to use the element in the document. Semantics will also determine where elements may be defined in the document hierarchy.

Examples of formatting text blocks based on semantics: A paragraph will create a block of text, grouping sentences to form a basic unspecific semantic unit. If you need to create a list of items, use a suitable list element, which will have a different meaning to paragraphs.

If you need a separate “highlighted” description of the topic or theme that the paragraphs or other content discuss, define a heading.

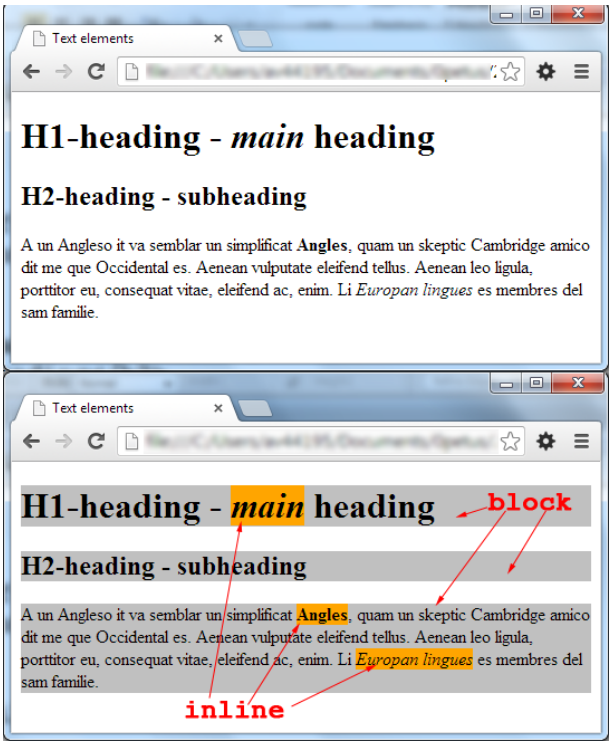
The image on the right shows block elements in the document flow. There two headings and a paragraph containing multiple sentences that are displayed in new separate blocks.



The presentation of the elements is not important, though these render differently due to differences in default browser styles. The default styles suggest certain differences in the meanings of the elements, such as the main heading is bigger than the subheading showing more importance etc. The visual differences may be further emphasized or mitigated with the choices of the designer by using CSS styles.

The above image also shows different styles for parts of the texts within the blocks. Some texts are in italics or bolded. These elements define meaning at **text-level** (**inline**) within block elements. Here the **strong** and **em** elements add more importance/emphasis to the selected contents within the parent blocks. Text-level elements do not start new lines by default.

The image on the right shows block and text-level elements, the later are in orange, the background of the blocks is highlighted with gray color.



For later reference the block and inline names are used here and in combination with CSS definitions. These reflect the nature how elements are rendered and not their semantic meaning.

There are some basic rules to guide you which elements are suited for what contents. Some elements are frequently used; some are rare and more specific in their meaning. Remember that element selection should be based on the meaning and possible role of the element within the document.

There are two generic elements, which do not have any particular semantic meaning. These are the block **div** and text-level **span**. These are typically used in situations where specific presentation is needed but there is no need to add or alter the semantics of the content. All elements may have their presentation altered very much freely by using CSS styles. Do not choose elements based on their default layout, but rather their semantic role in the document!

Next, some typical and frequently used elements are listed and explained. This list does not include all valid elements, only a selection. There are also new elements and features introduced with new versions of HTML.

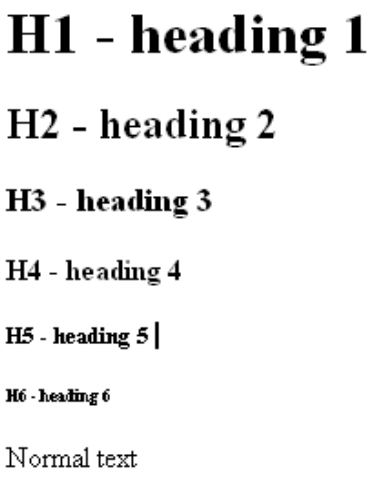
Block-level elements

Headings

Create document hierarchy with **headings**. Heading elements should briefly describe the topic or theme of a section of text it introduces. Heading elements are marked *<hn>...</hn>,* where n = 1-6, yielding 6 possible levels to define heading hierarchy in the text content. Headings can contain inline elements, but **not** block-level elements.

Use the headings in their hierarchical (semantic) order starting with **h1** – top level heading decreasing importance towards the lower levels.

Use h1 for the first or main headings, h2 for subheadings, h3 if you need more heading level depth, etc. The hierarchy is visually emphasised by default browser rendering styles, but the semantics is more important, styles can be adjusted. Having 6 levels, does not mean that you need to use all 6 in one document. Heading levels are not meant for defining numeric order for headings! The image on the right shows the default rendering of the heading levels in comparison to default text size.



Heading hierarchy example with two hierarchical heading levels:

```
<h1>Heading – main heading</h1>
<p>Some text dolor sit amet, adipiscing elit...</p>
<p>Some other paragraph consectetuer adipiscing elit...</p>
<h2>Second level heading 1</h2>
<p>Some text content consectetuer adipiscing elit.</p>
<h2>Second level heading 2</h2>
<p>Some text dolor sit amet, adipiscing elit...</p>
```

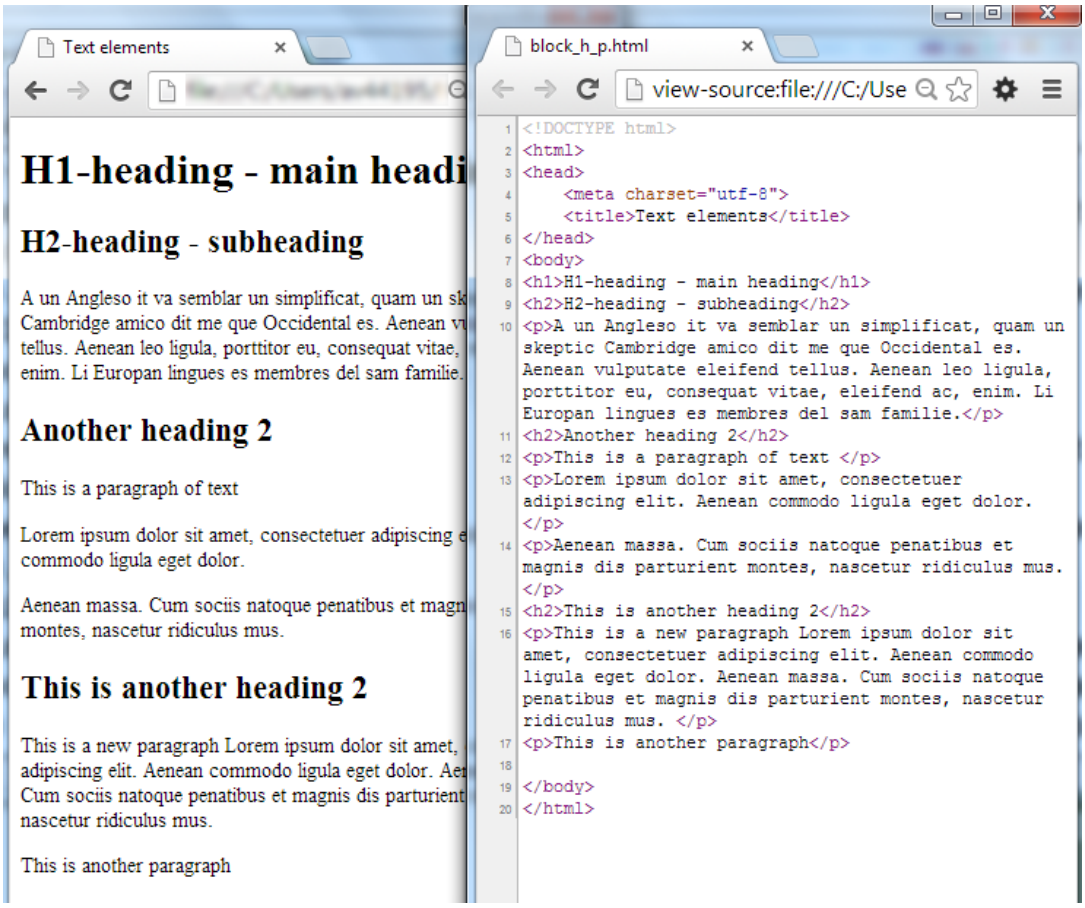
Headings cannot include other than **inline elements**, and images… other block type elements are not allowed!

HTML 5 allows you to define an outline for the document with heading elements and restart heading hierarchy in semantics sections of the document.

Paragraphs

Text body can be divided into simple **paragraph** blocks or text content like sentences can be grouped together with the * … * element. Paragraphs may contain text and possibly inline elements, cannot contain block-level elements (nor can headings). In a paragraph, text is **continuous, without breaks** (unless you add line-breaks or styles)**.** Each paragraph starts a new line (same as headings). The text lines will wrap as horizontal space runs out and continue on the next line! When the window gets resized the text is also resized.

Paragraphs and headings are the main and most generally used building blocks of textual page contents:



In some cases paragraphs may be replaced by more specific block elements if the content of the element requires a more specific semantic markup.

Lists

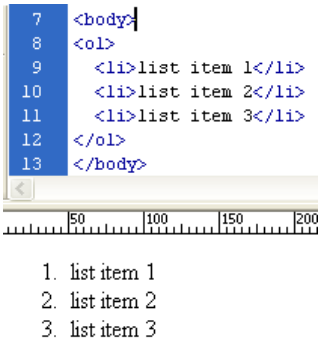
Text may be organized to lists. List elements are block elements and have a special hierarchy to define list items. List items reside in list container elements to form ordered or unordered lists, or a third type, a description list.

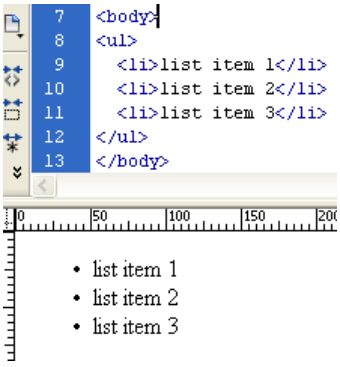
*
... *

- **Unordered list.**
- ...
- ...

*
... *

1. **Ordered list.**
2. ...
3. ...





Ul and ol elements cannot contain other elements **only li** elements as an immediate child (child = element on the next hierarchical level). The list elements cannot be within headings or paragraph elements.

Content of list items are placed within *
... elements.* Li elements can contain *plain text, images, block, and inline elements*, anything that is defining content in the body.

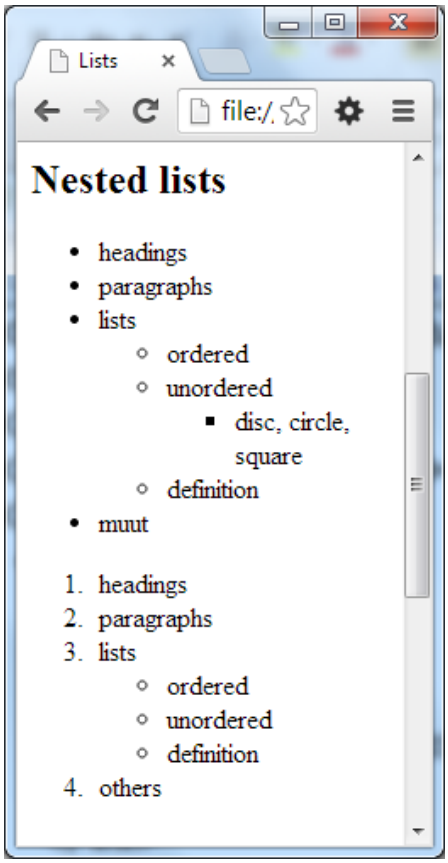
When you create a new ordered list, the order of numbering always starts from 1 by default. You may also change the starting number for the ordered lists manually with the **start** attribute – *<ol start="10">*

```
<ol>
<li>item 1</li>
…
<li>item 9</li>
</ol>

<p>some other content</p>

<ol start="10">
<li>numbering starts form 10</li>
…
</ol>
```

You can even reverse the numbering:

```
<ol start="10" reversed>
<li>numbering starts form 10</li>
<li>this woul be 9</li>
<li>this is 8</li>
…
<li>this is 0</li>
<li>this is -1</li>
</ol>
```

Lists may be also nested, lists within lists.



Lists embedded within another list **must be** in the parent list's list item - li element:

```
<ul>
 <li>ordered</li>
 <li>unordered
   <ul>
     <li>circle, square</li>
   </ul>
 </li>
</ul>
```

When you nest lists, remember to add the nested list within an LI element (as above)! You can also mix unordered and ordered lists when nested. Bullets and numbering can be controlled with CSS styles for the whole list or for individual items.

Description lists have a bit different structure.

<dl>

<dt> description or term*</dt>*
<dd> definition*</dd>…*

</dl>

The DL element is used to define the description list, DT elements for the terms to be defined, DD elements for the definition/description of the terms. DT may be followed by none or even more than one DD elements. There can be multiple DT elements described with DD descriptions as well. If you ***define*** *a term*, you should also include the <dfn> (definition) inline element for proper semantical structure.

http://html5doctor.com/the-dl-element/

```
<dl>
    <dt><dfn>HTML</dfn></dt>
    <dd>Hypertext Markup Language for defining document …</dd>
    <dd> Also refers to HTML5 here</dd>
    <dt><dfn>CSS<dfn></dt>
    <dd>Cascading Style Sheets for defining docume…</dd>
</dl>
```

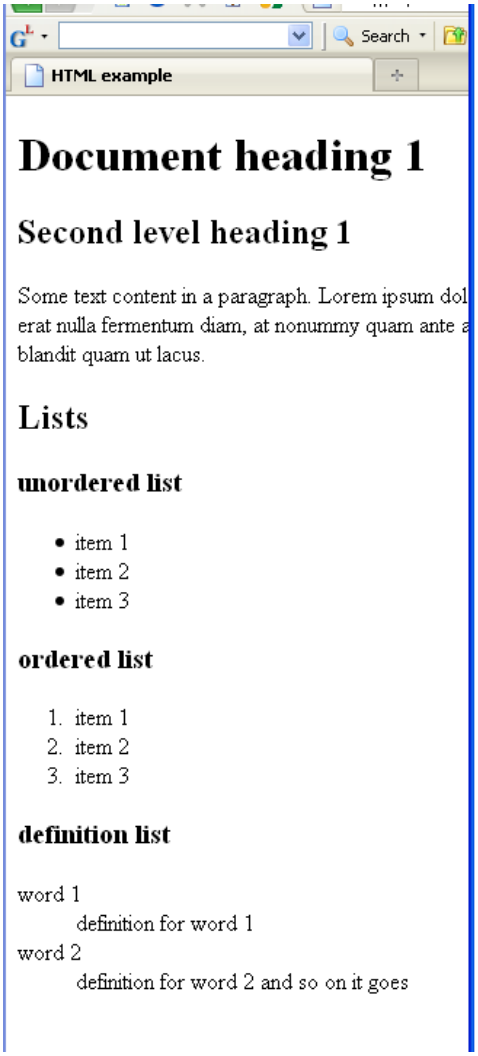
HTML

Hypertext Markup Language for defining document structure

Also refers to HTML5 here

CSS

Cascading Style Sheets for defining document layout



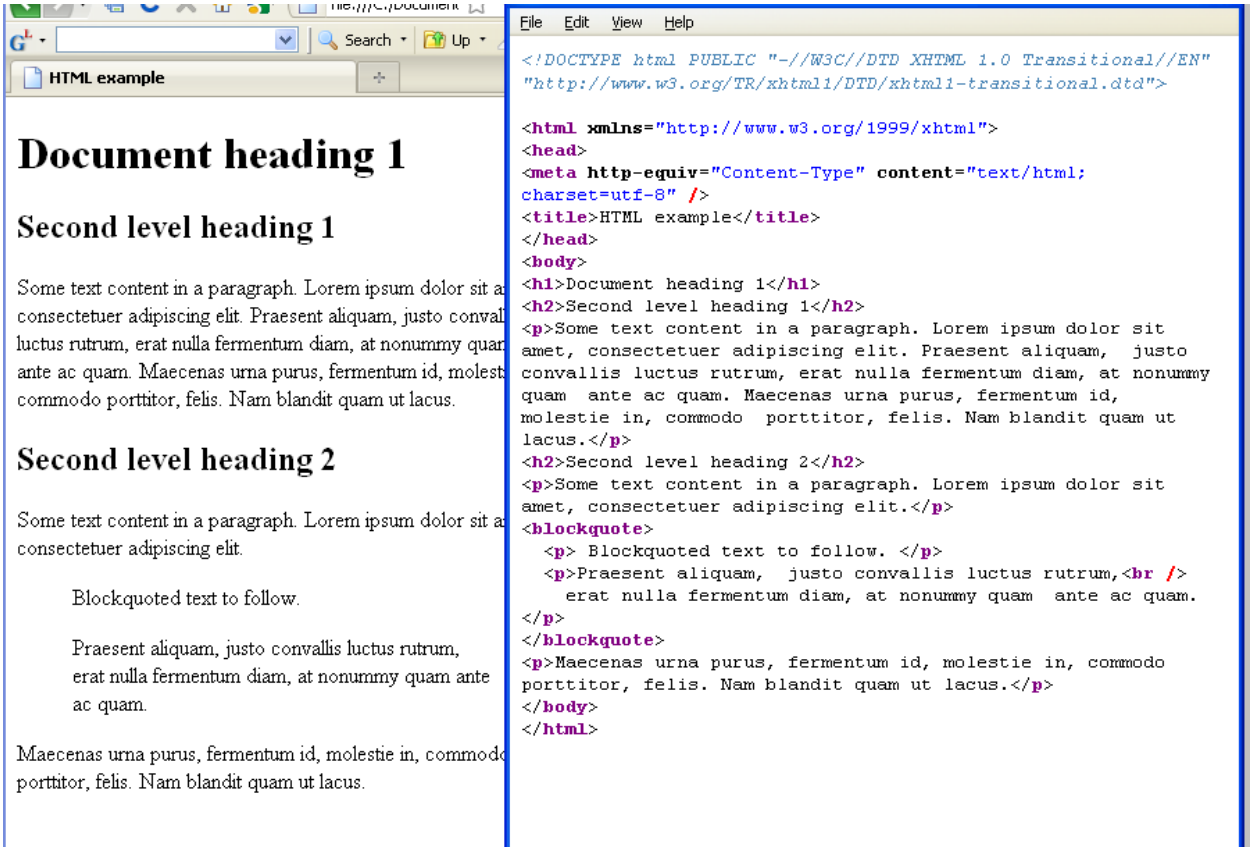

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
<title>HTML example</title>
</head>
<body>
<div id="pageheader">
<h1>Document heading 1</h1>
<h2>Second level heading 1</h2>
<p>Some text content in a paragraph. Lorem ipsum dolor sit
amet, consectetuer adipiscing elit. Praesent aliquam,  justo
convallis luctus rutrum, erat nulla fermentum diam, at nonummy
quam  ante ac quam. Maecenas urna purus, fermentum id,
molestie in, commodo  porttitor, felis. Nam blandit quam ut
lacus.</p>
</div>
<div id="examples">
<h2>Lists</h2>
<h3>unordered list</h3>
<ul>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
</ul>
<h3>ordered list</h3>
<ol>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
</ol>
<h3>definition list</h3>
<dl>
<dt>word 1</dt>
<dd>definition for word 1</dd>
<dt>word 2</dt>
<dd>definition for word 2 and so on it goes</dd>
</dl>
</div>
</body>
</html>
```

Quotations for blocks

<blockquote>…</blockquote>

Blockquotes are especially meant for longer quotations. This can be a quotation form a book, a document, a web page, etc. A blockquote element can contain texts like a single sentence or longer formatted texts spanning over several blocks with e.g. paragraphs and headings or other block-level and text-level elements. The default presentation is that the element content is indented both left and right margins.

http://html5doctor.com/cite-and-blockquote-reloaded/



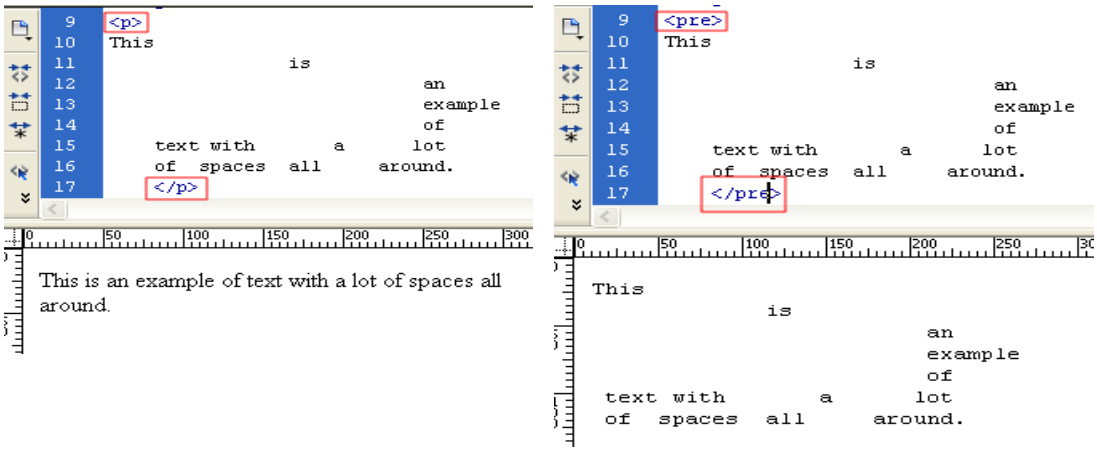
Shorter inline quotations can be marked up at the text-level with the **q** element. See text-level elements below.

Preformatted text

Normally the browser would disregard any visual text formatting in HTML documents. Extra spaces, line-breaks, tabulators are ignored. Text content may preserve its formatting when added within a *<pre>...</pre>* element.

Preserving formatting may have some use where spacing has a meaning e.g. in code examples, texts in a poem, etc. The pre element creates a block, where other blocks and images are not allowed, only text level elements. The text in the pre element is displayed with an equal-width font (monospace). Authors are discouraged to change the appearance of the preformatted text with CSS. PRE elements should not be used for longer texts and for styling blocks!

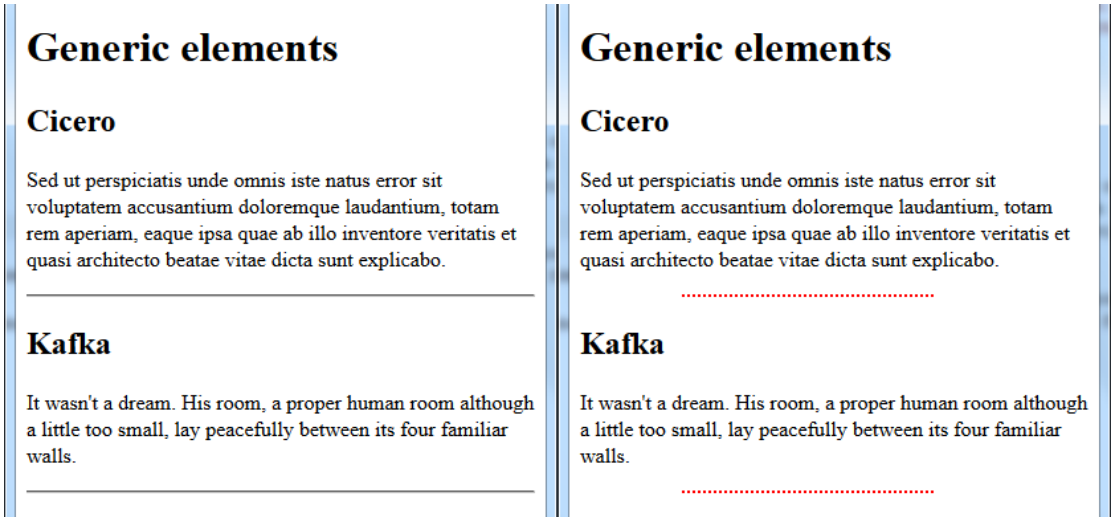
Below, a text with additional spacing in a paragraph (left) and in a pre element (right).



Horizontal rules – paragraph-level thematic break

This special block-level element has no content and it is used to separate thematically separate sections within a page. Typically placed to mark the change between themes and topics in the text of a document. **Hr** element traditionally adds a horizontal line to the web page. It is a block-level element that adds space before and after the line. You can use CSS to control its presentation (uses borders to create the line).

<hr> *(*in XHTML *<hr />)*



Other new elements in HTML 5

HTML5 also introduced several new elements to further describe block contents semantically within a page. These elements have no default layout (like the generic div element) and they are used for adding semantics, though you can style these elements too with CSS styles.

Some of the new elements are: section, article, main, header, footer, nav, sidebar, address, figure, figcaption. The course will not concentrate on these, but here are some simple descriptions about their intended use.

section – for thematic sectioning content in general, typically used with headings to create structured themes.

article – independent part of the document, an entity as an article of a newspaper, magazine…, can be used elsewhere e.g. as syndicated content (news article, blog writing, social media post…), heading structure is also advised.

main – defines the main content of the document, document specific part, e.g. the content apart from site-wise repeated structures like navigation, decorative elements, non-page specific content.

nav – main navigation of the site between pages or navigating within the page to different sections, this is not meant to mark up all the links in the document.

aside – a section containing related, but considered separate, content to contents in a section, like a sidebar for quotes, additional details, adds.

header and **footer** – introductory content or a closing part in a section or the page.

address – contact information for a section or page content

figure, figcaption – for annotating illustrations

https://www.w3.org/TR/html/sections.html#sections

https://www.w3.org/TR/html/grouping-content.html#grouping-content

https://developer.mozilla.org/en-US/docs/Web/HTML/Element

http://html5doctor.com/element-index/

Inline or text-level elements

also referred as Phrasing elements

Text-level phrasing elements all have a separate semantic meaning too and presentation is NOT important! There are references to default layout features, but these can be redesigned and use a different presentation. Inline elements can add additional meaning to parts of the text content within block elements. They may be also used in other inline elements. Semantics can be further refined with selected attributes and values.

... **Stress emphasis** added (usually the text is rendered in *italics*)

... for separating texts that have a different tone from surrounding texts e.g. technical term, a phrase in a different language, a thought… (in italics) **alternate voice**

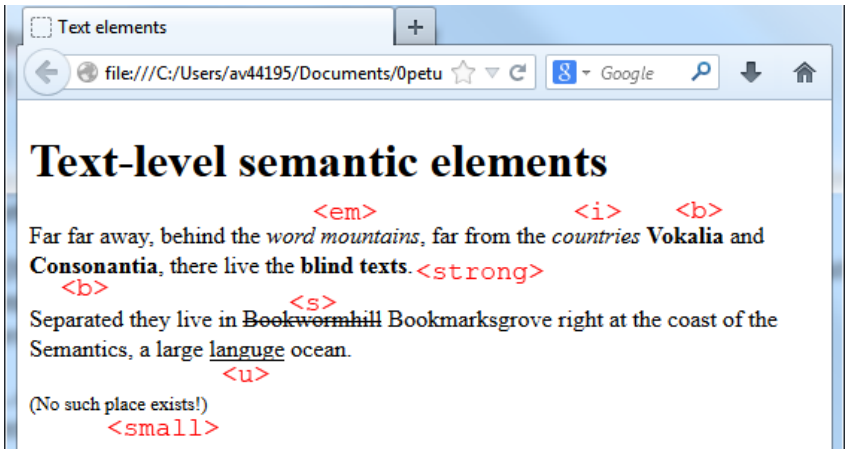
… strong **importance** of the enclosed text, renders as bold text

... to separate **keyword**, product names, phrases but without adding special importance

<s>...</s> marks content that is no longer relevant, accurate or applicable (strikethrough) – **inaccurate text**

<u>...</u> **annotations** (underline) can mark misspelled texts

`<small>…</small>` for small print, adding a sidenote, visible comment




```
<p>Far far away, behind the <em>word mountains</em>, far from the
<i>countries</i> <b>Vokalia</b> and <b>Consonantia</b>, there live the
<strong>blind texts</strong>.</p>

<p>Separated they live in <s>Bookwormhill</s> Bookmarksgrove right at the
coast of the Semantics, a large <u>languge</u> ocean.</p><p><small>(No such
place exists!)</small></p>
```

`<q>…</q>` for short **quotes** at text-level – current browsers add quotation marks automatically.

`<abbr>…</abbr>` - abbreviations and acronyms

```
<abbr title="World Wide Web">WWW</abbr>
<abbr title="exempli gratia, for example">e.g.</abbr>
```

Here the ABBR element may include a **title** attribute that shows on mouse-over the origin of the abbreviation.

`<cite>…</cite>` Citations, references to **titles** of other texts, books – text rendered in italics

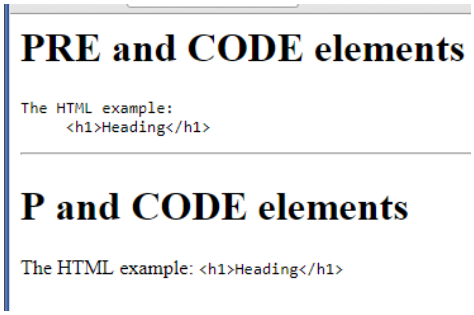
```
<p>A great book on web publising: <cite>Learning web design</cite> by Jennifer Niederst Robbins</p>
```

`<dfn>…</dfn>` a defining (important) term in the document, see also description lists

```
<p><dfn>block elements</dfn> like paragraphs and headings…</p>
```

`<mark>…</mark>` to **highlight** text in the context (default yellow background)

`<code>…</code>` For **code samples** at text-level. Text within the tag is rendered in `monospace` font style like in pre elements. The code element can also mark-up example code within pre elements where you can maintain text formatting and spacing in the code, the p element does not maintain spacing.




```
<pre>The HTML example:
    <code>&amp;lt;h1&amp;gt;Heading&amp;lt;/h1&amp;gt;</code>
</pre>
```

`<sub>…</sub>` marks text as **subscript**

`<sup>…</sup>` marks text as **superscript**

more examples...

`<q>`

Short quotation: And he said "don't panic just yet" with a smile on his face.

`<abbr>`

Abbreviation and acronym: WWW or e.g. as examples, hover your mouse pointer over them to see what they mean.

`<cite>`

A great book on web publising: *Learning web design* by Jennifer Niederst Robbins

`<dfn>`

block elements like paragraphs and headings…

`<mark>`

A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart.

Code examples:

`<code>`

A text paragraph in HTML is amrked with the `<p>...</p>` code.

`<sup>`

24-bit gives 2^{24} possible values

`<sub>`

water is H_2O

Line-break and word-break

To create a line-break within a e.g. paragraph or block of text, use *
 (XHTML <br />)* element to end the current line of text at the insertion of the tag. **No closing tag (void element)!** No extra gap between lines of texts. The break does not terminate the paragraph or other blocks, only the line! The line-break must not be used outside texts nor to only create gaps for presentational purposes!

You can also define a point where a log word can be broken to parts. For this you may add the **WBR** (word-break) element. <wbr> has no closing tag either. The browser breaks the word if the word cannot be fitted to the available horizontal space.

There are many other phrasing elements; this is a short list of typically used options.

https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Inline_text_semantics

Summary of text-level phrasing elements

https://www.w3.org/TR/html/textlevel-semantics.html#text-level-semantics-usage-summary

https://www.w3.org/TR/html/textlevel-semantics.html#textlevel-semantics

Generic elements with no real semantical meaning

DIV and SPAN

The generic elements provide a way to create custom elements without any real semantic meaning. These elements are meant to be used for defining visual presentation with CSS styles.

The ***div*** element can be used to indicate generic **block-level** structures to divide or group text into identifiable blocks and create presentation for these.

```
<div id="sectionl">
<p>Lorem ipsum dolor sit amet,
consetetur sadipscing elitr.</p>
<p>Sed diam nonumy eirmod tempor
invidunt ut labore et dolore magna
aliquyam erat, sed diam voluptua</p>
</div>
```

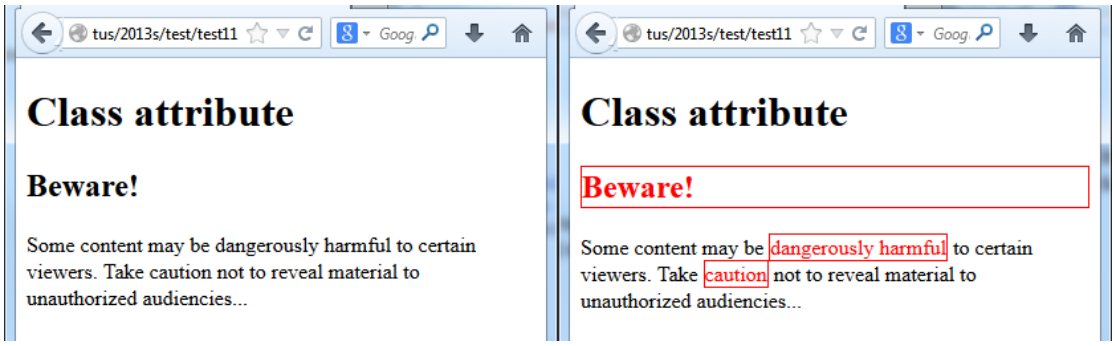
The ***span*** element indicates generic **inline** text-level elements that may be used within e.g. block-level elements.

```
<p>Lorem ipsum dolor sit amet, Sed diam nonumy eirmod
temporconsetetur <span class="red">sadipscing</span> Sed
diam nonumy eirmod tempor elitr.</p>
<p>Sed diam nonumy eirmod tempor invidunt ut labore et
```

To add layout with style rules, generic elements are typically identified or grouped with the **id** and **class** attributes. The id attribute is for giving a unique name to an element, class is for creating a group of elements that share some common presentational style properties. Both attributes may be used with other elements too.

```
<h2 class="alert">Beware!</h2>

<p>Some content may be <span class="alert">dangerously harmful</span>
to certain viewers. Take <strong class="alert">caution</strong> not
to reveal material to unauthorized audiencies...</p>
```

class attribute to group element - h2, span, & strong to share common layout properties.



Nesting HTML elements and other syntax examples

Nesting means that HTML elements can be contained within other elements. No partial overlapping is allowed. In the example starting the title after the head requires that the title should be closed before the head. There are elements that cannot be contained by certain other elements (hierarchy)…

<head><title>... nesting example … *</title></head>* => **correct**

<head><title>... nesting example … ***</head></title>*** => **incorrect !!! (**wrong closing order)

<title>*<head>...* nesting example … *</head>****</title>*** => **incorrect !!!**
(wrong order <title> must be within the <head>!)

End tag should match starting tag:

<h1>... nesting example … ~~***</h2>***~~ => **incorrect !!!**

Some elements must be nested within certain elements

(here <li> within <ol> or <ul>):

<ol>
*
ordered list item
*
*
ordered list item
*
*
*

<ul>
~~*<h1>heading</h1>*~~ (h1 element is not within an li element or outside the ul)
*
item 1
*
*
item 2
*
*
*

<ul>
*
<h1>heading</h1>
*
*
item 1
*
*
item 2
*
*
*

<h1>heading</h1>
<ul>
*
item 1
*
*
item 2
*
*
*

Some elements may not be nested in others, like:

~~<h1>~~Heading ~~</h1>~~Some text content…

<h1>~~ ~~Some strange idea~~ ~~</h1>

This is correct; div can group other block elements (also without semantic meaning) or see e.g. blockquote, section and the like:

```
<div>
      <h1>heading text</h1>
      <p>Some other text</p>
</div>
```

but this is incorrect, you cannot group text blocks with a p element

~~ ~~
<ul>
*
item 1
*
*
item 2
*
*
*
~~ ~~

instead of a p element, you could have a generic div element… …

This would be incorrect:

<ol>
*
item 1
*
*
item 2
*
~~*<h1>Another section</h1>*~~
~~* more text *~~
*
item 3
*
*
item 4
*
*
*
You cannot leave the list open e.g. to keep the numbering running in an ol list!
Instead:

<ol>
*
item 1
*
*
item 2
*
*
*
<h1>Another section</h1>
* more text *
<ol start="3">
*
item 3
*
*
item 4
*
*
*

Elements should be opened and closed according to hierarchical rules

<head><title>... nesting example … *</title></head>*

* *Some text… ** - incorrect order!!!**
* *Some text… ** - incorrect closing order!!!**
* *Some text… ** - correct**
P is a block element that may contain text-level elements (e.g. STRONG) and not vice-versa.

 Emphasis and importance **- incorrect !!!**

 Emphasis and importance **- incorrect !!!**

 Emphasis and importance **- correct**

The element should close before the element close (starting order) and both are within the paragraph element!

Many types of block-level elements, like headings, paragraphs, pre cannot contain other blocks.

Elements like div, li and blockquote (+ HTML5 sectioning elements) may contain other blocks.

This would be also wrong:

```
<p>This is a code example:
 <pre>The HTML example:
      <code><h1>Heading</h1></code>
 </pre>
</p>
```

cannot nest p and pre, also the < and > characters are reserved for HTML markup!

Rather it should be:

```
<p>This is a code example:</p>
<pre>The HTML example:
      <code>&lt;h1&gt;Heading&lt;/h1&gt;</code>
</pre>
```

Another approach would be much less optimal than the above; the p element does not preserve spacing, the two examples would also be different in rendering:

```
<p>This is a code example:<br>
The HTML example:<br>
      <code>&lt;h1&gt;Heading&lt;/h1&gt;</code>
<p>
```

Also remember that **attributes** are defined in the element's **opening tag**, never in the end tag!

```
<h1 class="first">Main heading – topic 2</h1>
```

Also, some elements **must not** have a closing tag:

```
<meta></meta>
<link></link>
<br></br>
<wbr></wbr>
<mig></img>
```

but may be closed as in XHTML <meta /> <link />
 <wbr /> <img />…