

# HTML5

## What is it?

They lean on the web, but it gives way; they cling to it, but it does not hold.

[Job 8:15 NIV](#)

HTML5 is an abbreviation that stands for **H**yper **T**ext **M**arkup **L**anguage version **5**.

It is an extension to XML; e**X**tensible **M**arkup **L**anguage.

HTML was invented by Tim Berners-Lee ([@timbl](#) - <https://github.com/timbl>) in 1989 while he worked for [CERN](#) - <http://home.web.cern.ch/>. The early versions of HTML was based on SGML (Standard Generalized Mark-up Language), and so they were not XML.

## Why should I care?

HTML5 is what makes up web pages, and since a *Web Application* consists of web pages, we need to know how to create HTML5 content.

Let's look at XML syntax first.

An XML document is a file, that contains structured information. The structure is defined by using the following characters that have special meaning in XML:

< > / " = & # ;

# HTML5

## Details

### Elements

The < and > characters are used to define elements.

An element can either have content, or be empty.

An empty element must start with < and end with />. In between is the name of the element and maybe some attributes, we'll get back to those.

```
<html />
```

The word html is the name of the element. An element in XML can have any name.

An element with content must first have a start-tag that starts with < and ends with >, then comes the content, which can be text, or other elements, and then at the end comes the end-tag, which starts with </ and ends with >.

```
<html>
```

```
    Some content
```

```
</html>
```

An element can have any number of child-elements, which can again have child-elements to any level of nesting you want.

```
<html>
```

```
  <head>
```

```
    <title>My first web page</title>
```

```
  </head>
```

```
  <body>
```

```
    <p>Here is a paragraph</p>
```

```
  </body>
```

```
</html>
```

# HTML5

It is customary to indent child-elements, but it is not necessary. It just makes it more pretty, and easier to read.

Something like this is perfectly legal XML syntax:

```
<html><body><p>Here is a paragraph</p></body></html>
```

## Structure of a Web Page

An HTML5 file must always start with:

```
<!DOCTYPE html>
```

This tells the browser that what follows is HTML5.

Next comes the `html` element. It does not need to have any child-elements, so this is a perfectly legal HTML5 document:

```
<!DOCTYPE html>
<html/>
```

The first child-element of the `html` element is the `head` element.

The `head` element contains things like `title`, `meta`, `script`, `style`, `link` etc.

Example:

```
<head>
  <title>My First Web Page</title>
  <meta charset="utf-8" />
</head>
```

The `body` element is the next child-element of the `html` element. It contains the content of the document.

Example:

# HTML5

```
<body>
  <h1>This is the biggest header</h1>
  <h2>This is smaller</h2>
  <p>This is a paragraph</p>
</body>
```

## Web addresses

A web address or a so called URL (**U**niform **R**esource **L**ocator) is used to indicate the location of a web page, an image, or some other resource.

A URL can be relative, or absolute. Absolute URLs always contain a scheme. A relative URL is relative to a base URL.

Examples:

Absolute: <https://github.com/mscsbend/Chromium-Course/wiki/HTML5>

Relative: </mscsbend/Chromium-Course/wiki/HTML5>

Relative: [CSS](#)

Relative: [../wiki](#)

Please note that even though one of the URLs above starts with a slash (/), it is still a relative URL, but it has an absolute path.

The parts of a URL:

scheme:scheme-specific-part

The parts of an HTTP URL:

# HTML5

`http://hostname:port/path?query#fragment`

The scheme can be either `http`, or `https`. The `https` scheme indicates that the connection is secured and encrypted.

The `hostname` is the name of the computer that the web resource is located on such as `github.com`.

The `port` is the TCP port number. It basically means the channel the communication goes on. The default value for the `http` scheme is 80, and if that is used, it does not need to be specified. Default for the `https` scheme is 443.

The `path` is the location in the file system of the host computer. Usually it maps to be relative to a so called document root of the web site, but not necessarily.

The `query` is extra parameters that the web resource may need. It may specify how to render the content.

The `fragment` points to an anchor inside the resource.

## Headings

The elements `h1`, `h2`, `h3`, `h4`, `h5`, and `h6` are headings. The higher the number, the smaller the heading.

You will use these elements to mark sections and chapters in your document. Usually you will expect to be able to link directly to one of these elements by appending some hash-tag (`#`) to the end of the web address to get to the *fragment*. And yes, hash-tags were **not** invented by Twitter. In HTML lingo a hash-tag is called a pound-sign.

Actually the hash-tag does not point to an `h*` element, but rather to

# HTML5

an `a` element, that will usually be placed right inside the `h*` element.

Example:

```
<h1><a name="top" />The top heading</h1>
```

The address to go straight to "The top heading" would be:

`http://www.some-domain.com/some-path/some-page.html#top`

## Links

The `a` element is used to either indicate a link to somewhere, or as the destination of a link. In HTML5 lingo that is called an anchor.

Use it as an anchor by specifying the `name` attribute. To navigate to an anchor in a page, append the hash-tag (#) followed by the value of the `name` attribute to the address of the page. An anchor is usually invisible and so the element will be left empty; without content.

Example:

```
<a name="top" />
```

To use the `a` element as a link, specify the `href` attribute. The `href` attribute value is the web address of another page, or it could be inside the same page by just specifying the hash-tag followed by the name of an anchor in the page. The content of the `a` element will be the text that the user can click on. A link is usually (but not always) displayed in blue underlined font.

Example:

```
<a href="#top">Go to the top</a>
<a href="http://github.com">Visit GitHub</a>
```

# HTML5

## Images

The `img` element is used to show images in the web page.

The `src` attribute indicates the location of the image as a URL.

Example:

```

```

This result in the following image being inserted in the web page:



An image can be used as a link instead of text. To make this effect, just add the `img` element in the content of the `a` element like this:

```
<a href="https://github.com">  
    
</a>
```

# HTML5

## Lists

Lists can be *ordered* or numbered like this:

1. First item
2. Second item

Or they can be *unordered* or bulleted like this:

- Some item
- Another item

The *ordered* list is specified using the `ol` element and the *unordered* list by specifying the `ul` element. Each list item is specified with the `li` element.

Here is the HTML for the two examples above:

```
<ol>
  <li>First item</li>
  <li>Second item</li>
</ol>

<ul>
  <li>Some item</li>
  <li>Another item</li>
</ul>
```

Lists can also be nested like this:

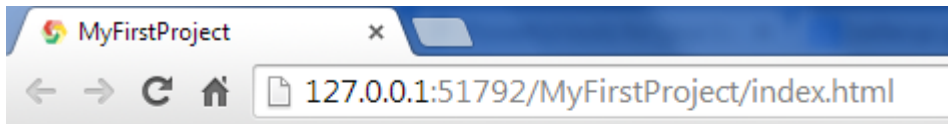
```
<ol>
  <li>First item in the top level</li>
  <ol>
    <li>Second level item</li>
```



# HTML5

```
<li>Another second level item</li>
</ol>
<li>Second item in the top level</li>
</ol>
```

It shows like this in the web page:



1. First item in the top level
  1. Second level item
  2. Another second level item
2. Second item in the top level

## Tables

Tables lets you put web page content into rows and columns. Each table is defined by a `table` element.

Each row in the table is defined using the `tr` element inside the `table` element. Inside the `tr` element, each cell is defined using the `td` element. The content of the `td` element can be text or other HTML elements such as `a`, `img`, etc.

Example:

```
<table>
  <tr>
    <td>first row, first cell</td>
    <td>first row, second cell</td>
  </tr>
  <tr>
```

# HTML5

```
<td>second row, first cell</td>
<td>second row, second cell</td>
</tr>
</table>
```

Renders like this:

|                        |                         |
|------------------------|-------------------------|
| first row, first cell  | first row, second cell  |
| second row, first cell | second row, second cell |

If we add a bit of styling it is easier to see where the cells are - more about that in the next chapter.

|                        |                         |
|------------------------|-------------------------|
| first row, first cell  | first row, second cell  |
| second row, first cell | second row, second cell |

You can define headings using the `th` element.

Example:

```
<table>
  <tr>
    <th>heading row, first cell</th>
    <th>heading row, second cell</th>
  </tr>
  <tr>
    <td>second row, first cell</td>
    <td>second row, second cell</td>
  </tr>
</table>
```

Renders like this:

| heading row, first cell | heading row, second cell |
|-------------------------|--------------------------|
| second row, first cell  | second row, second cell  |

# HTML5

Cells can be joined either horizontally using the `colspan` attribute or vertically using the `rowspan` attribute.

Example:

```
<table>
  <tr>
    <th colspan="2">heading row, first cell</th>
    <th colspan="2">heading row, second cell</th>
  </tr>
  <tr>
    <td rowspan="2" colspan="2">second row, first cell</td>
    <td rowspan="2">second row, second cell</td>
    <td>second row, third cell</td>
  </tr>
  <tr>
    <td>second row, first cell</td>
  </tr>
  <tr>
    <td>third row, first cell</td>
    <td>third row, second cell</td>
    <td>third row, third cell</td>
    <td>third row, fourth cell</td>
  </tr>
</table>
```

Renders like this:

| heading row, first cell |                        | heading row, second cell |                        |
|-------------------------|------------------------|--------------------------|------------------------|
| second row, first cell  |                        | second row, second cell  | second row, third cell |
|                         |                        |                          | second row, first cell |
| third row, first cell   | third row, second cell | third row, third cell    | third row, fourth cell |

# HTML5

## Forms

Forms are used to let the user enter information to be sent to a server, or to be used by the page.

The `input` element is used to input most of the data in a form.

The `type` attribute indicates how the data is to be entered. If the `type` attribute has the value `text`, the input element is used to enter a simple text in one line.

Example:

```
<form>
  <label for="name">Name</label>
  <input id="name" name="name" type="text" /><br/><br/>
  <label for="age">Age</label>
  <input id="age" name="age" type="text" />
</form>
```

Name

Age

The `label` element shows what information is expected to be entered.

The `for` attribute points to the `id` attribute of an `input` element, so we can see what it is a label for.

The `name` attribute is used by the server when the form is received. It is what identifies the input.

With the `value` attribute, you can fill in the `input` element like this:

```
<form>
  <label for="name">Name</label>
  <input id="name" name="name" type="text" value="Palle Cogburn"
/><br/><br/>
```

# HTML5

```
<label for="age">Age</label>
<input id="age" name="age" type="text" value="47" />
</form>
```

Name

Age

The placeholder attribute is used to give the user information about what to enter. This information is cleared as soon as the user starts entering data.

```
<form>
  <label for="name">Name</label>
  <input id="name" name="name" type="text"
    placeholder="Enter your name" /><br/><br/>
  <label for="age">Age</label>
  <input id="age" name="age" type="text"
    placeholder="How old are you?" />
</form>
```

Name


Age

You can show a drop-down list of options with the select element.

```
<label for="month">Birth month</label>
<select id="month" name="month">
  <option value="1">January</option>
  <option value="2">February</option>
  <option value="3">March</option>
  <option value="4">April</option>
  <option value="5">May</option>
  <option value="6">June</option>
  <option value="7">July</option>
```

# HTML5

```
<option value="8">August</option>
<option value="9">September</option>
<option value="10">October</option>
<option value="11">November</option>
<option value="12">December</option>
</select>
```

Birth month 

Inside the select element, each option element specifies the potential value of the select element in the value attribute.

If there are only two options, using a checkbox is the best solution.

The type attribute value of checkbox in the input element is used for this.

```
<label for="married">Married</label>
<input id="married" name="married" type="checkbox" checked />
```

Married ☒

Notice how there is a checked attribute in the input element. If it is there, the checkbox is checked, otherwise it is unchecked.

Usually there will be a button on the form to submit the information when you are done filling it out. This is an input element with the type attribute value of submit.

# HTML5

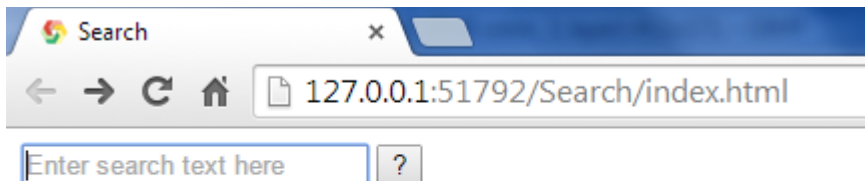
```
<input type="submit" value="Submit" />
```

A rectangular button with a light gray border and the word "Submit" in a dark gray sans-serif font.

The action attribute on the `form` element is used to tell the browser the web address where the form data should be sent to when the submit button is clicked.

Here is a web page where you can search on Google:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Search</title>
  </head>
  <body>
    <form action="http://google.com/search">
      <input name="q" type="text" placeholder="Enter search text
here" />
      <input type="submit" value="?" />
    </form>
  </body>
</html>
```



When you type in some text to search for; "mscsbend" in the example, and then press the `<Enter>` key or click the button, you will see the result from

# HTML5

Google in the browser. Try  
it!

Follow Us on Twitter [#MSCSBend](#) ... a Savior, which is Christ the LORD  
Educationally yours, . Joe Bales. Administrator. [joe.bales@mscsbend.org](mailto:joe.bales@mscsbend.org)  
[Google+ page](#) · [Be the first to review](#)

Notice that the `input` element with the `name` attribute equal to `q` now appears  
in the URL query part with the value you entered.

([https://www.google.com/search?q=mscsbend&gws\\_rd=ssl](https://www.google.com/search?q=mscsbend&gws_rd=ssl))

There are other elements you can put on forms, and you can explore these on  
your own here:

[http://www.w3schools.com/html/html\\_forms.asp](http://www.w3schools.com/html/html_forms.asp)

## Where can I find more information?

HTML Tutorial (<http://www.w3schools.com/html/>)

HTML (Hyper Text Markup Language | MDN (<https://developer.mozilla.org/en->



# HTML5

[US/docs/Web/HTML](#))

[HTML5 \(http://www.w3.org/TR/html5/\)](http://www.w3.org/TR/html5/)

## Challenges

1. Create a web page with a table of some of your friends. The table could contain first name, last name, birthday, phone number, etc.
2. Create a web page with pictures, as a photo album.
3. Create a web page with a form that has different types of input. An example could be a form where you fill in your personal information.