

# What is New in HTML5?

The DOCTYPE declaration for HTML5 is very simple:

```
<!DOCTYPE html>
```

The character encoding (charset) declaration is also very simple:

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

## HTML5 Example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document.....
</body>

</html>
```

The default character encoding in HTML5 is UTF-8.

# New HTML5 Elements

The most interesting new HTML5 elements are:

New **semantic elements** like <header>, <footer>, <article>, and <section>.

New **attributes of form elements** like number, date, time, calendar, and range.

New **graphic elements**: <svg> and <canvas>.

New **multimedia elements**: <audio> and <video>.

In the next chapter, [HTML5 Support](#), you will learn how to "teach" older browsers to handle "unknown" (new) HTML elements.

# New HTML5 API's (Application Programming Interfaces)

The most interesting new API's in HTML5 are:

- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage
- HTML Application Cache
- HTML Web Workers
- HTML SSE

**Tip:** HTML Local storage is a powerful replacement for cookies.

## Removed Elements in HTML5

The following HTML4 elements have been removed in HTML5:

Removed Element	Use Instead
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
<center>	CSS
<dir>	<ul>
<font>	CSS
<frame>	

<frameset>	
<noframes>	
<strike>	CSS, <s>, or <del>
<tt>	CSS

In the chapter [HTML5 Migration](#), you will learn how to easily migrate from HTML4 to HTML5.

## HTML History

Since the early days of the World Wide Web, there have been many versions of HTML:

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft

2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1

From 1991 to 1999, HTML developed from version 1 to version 4.

In year 2000, the World Wide Web Consortium (W3C) recommended XHTML 1.0. The XHTML syntax was strict, and the developers were forced to write valid and "well-formed" code.

In 2004, W3C's decided to close down the development of HTML, in favor of XHTML.

In 2004, WHATWG (Web Hypertext Application Technology Working Group) was formed. The WHATWG wanted to develop HTML, consistent with how the web was used, while being backward compatible with older versions of HTML.

In 2004 - 2006, the WHATWG gained support by the major browser vendors.

In 2006, W3C announced that they would support WHATWG.

In 2008, the first HTML5 public draft was released.

In 2012, WHATWG and W3C decided on a separation:

**WHATWG wanted to develop HTML as a "Living Standard"**. A living standard is always updated and improved. New features can be added, but old functionality cannot be removed.

The [WHATWG HTML5 Living Standard](#) was published in 2012, and is continuously updated.

**W3C wanted to develop a definitive HTML5 and XHTML standard.**

The [W3C HTML5 recommendation](#) was released 28 October 2014.

W3C also published an [HTML 5.1 Candidate Recommendation](#) on 21 June 2016.

## New Elements in HTML5

Below is a list of the new HTML5 elements, and a description of what they are used for.

## New Semantic/Structural Elements

HTML5 offers new elements for better document structure:

Tag	Description
<article>	Defines an article in a document
<aside>	Defines content aside from the page content
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<figcaption>	Defines a caption for a <figure> element
<figure>	Defines self-contained content
<footer>	Defines a footer for a document or section
<header>	Defines a header for a document or section
<main>	Defines the main content of a document
<mark>	Defines marked/highlighted text
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links

<progress>	Represents the progress of a task
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time
<wbr>	Defines a possible line-break

# HTML5 Semantic Elements


Semantics is the study of the meanings of words and phrases in a language.

Semantic elements = elements with a meaning.

## What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer. Examples of **non-semantic** elements: `<div>` and `<span>` - Tells nothing about its content. Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

## Browser Support

				
Yes	Yes	Yes	Yes	Yes

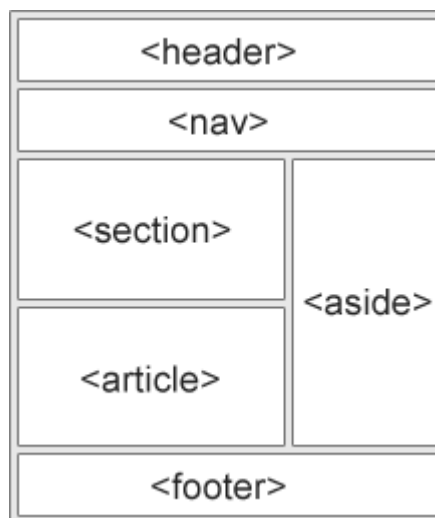
HTML5 semantic elements are supported in all modern browsers. In addition, you can "teach" older browsers how to handle "unknown elements".

## New Semantic Elements in HTML5

Many web sites contain HTML code like: `<div id="nav">` `<div class="header">` `<div id="footer">` to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`
- `<figure>`
- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<section>`
- `<summary>`
- `<time>`



# HTML5 <section> Element

The <section> element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

## Example

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is...</p>
</section>
```

# HTML5 <article> Element

The <article> element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

- Forum post
- Blog post
- Newspaper article

## Example

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

# Nesting <article> in <section> or Vice Versa?

The <article> element specifies independent, self-contained content.

The <section> element defines section in a document.

Can we use the definitions to decide how to nest those elements? No, we cannot!

So, on the Internet, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <sections> elements.

You will also find pages with <section> elements containing <section> elements, and <article> elements containing <article> elements.



Example for a newspaper: The sport **articles** in the sport **section**, may have a technical **section** in each **article**.

## HTML5 <header> Element

The <header> element specifies a header for a document or section.

The <header> element should be used as a container for introductory content.

You can have several <header> elements in one document.

The following example defines a header for an article:

### Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

## HTML5 <footer> Element

The <footer> element specifies a footer for a document or section.

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You may have several <footer> elements in one document.

### Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

## HTML5 <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

### Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

## HTML5 <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The aside content should be related to the surrounding content.

### Example

```
<p>My family and I visited The Epcot center this summer.</p>

<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

## HTML5 <figure> and <figcaption> Elements

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an image and a caption can be grouped together in a **<figure>** element:

### Example

```
<figure>
  
  <figcaption>Fig1. - The Pulpit Rock, Norway.</figcaption>
</figure>
```

The **<img>** element defines the image, the **<figcaption>** element defines the caption.

## Why Semantic Elements?

With HTML4, developers used their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.

This made it impossible for search engines to identify the correct web page content.

With the new HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier.

According to the W3C, a Semantic Web: "Allows data to be shared and reused across applications, enterprises, and communities."

# Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5.

Tag	Description
<a href="#"><u>&lt;article&gt;</u></a>	Defines an article
<a href="#"><u>&lt;aside&gt;</u></a>	Defines content aside from the page content
<a href="#"><u>&lt;details&gt;</u></a>	Defines additional details that the user can view or hide
<a href="#"><u>&lt;figcaption&gt;</u></a>	Defines a caption for a <figure> element
<a href="#"><u>&lt;figure&gt;</u></a>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<a href="#"><u>&lt;footer&gt;</u></a>	Defines a footer for a document or section
<a href="#"><u>&lt;header&gt;</u></a>	Specifies a header for a document or section
<a href="#"><u>&lt;main&gt;</u></a>	Specifies the main content of a document
<a href="#"><u>&lt;mark&gt;</u></a>	Defines marked/highlighted text
<a href="#"><u>&lt;nav&gt;</u></a>	Defines navigation links
<a href="#"><u>&lt;section&gt;</u></a>	Defines a section in a document
<a href="#"><u>&lt;summary&gt;</u></a>	Defines a visible heading for a <details> element

[<time>](#)

Defines a date/time

## Migration from HTML4 to HTML5

This chapter is entirely about how to **migrate** from **HTML4** to **HTML5**.

This chapter demonstrates how to convert an HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

You can migrate from XHTML to HTML5, using the same recipe.

Typical HTML4	Typical HTML5
<code>&lt;div id="header"&gt;</code>	<code>&lt;header&gt;</code>
<code>&lt;div id="menu"&gt;</code>	<code>&lt;nav&gt;</code>
<code>&lt;div id="content"&gt;</code>	<code>&lt;section&gt;</code>
<code>&lt;div class="article"&gt;</code>	<code>&lt;article&gt;</code>
<code>&lt;div id="footer"&gt;</code>	<code>&lt;footer&gt;</code>

## Change to HTML5 Doctype

Change the **doctype**:

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

to the HTML5 doctype:

### Example

```
<!DOCTYPE html>
```

# Change to HTML5 Encoding

Change the **encoding** information:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

to HTML5 encoding:

## Example

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

# Add The HTML5Shiv

The new HTML5 semantic elements are supported in all modern browsers. In addition, you can "teach" older browsers how to handle "unknown elements".

However, IE8 and earlier, does not allow styling of unknown elements. So, the HTML5Shiv is a JavaScript workaround to enable styling of HTML5 elements in versions of Internet Explorer prior to version 9.

Add the HTML5Shiv:

## Example

```
<!--[if lt IE 9]>  
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
<![endif]-->
```

# The Difference Between <article> <section> and <div>

There is a confusing (lack of) difference in the HTML5 standard, between <article> <section> and <div>.

In the HTML5 standard, the <section> element is defined as a block of related elements.

The <article> element is defined as a complete, self-contained block of related elements.

The <div> element is defined as a block of children elements.

How to interpret that?

In the example above, we have used <section> as a container for related <articles>.

But, we could have used <article> as a container for articles as well.

Here are some different examples:

<article> in <article>:

```
<article>
```

```
<h2>Famous Cities</h2>
```

```
<article>
```

```
<h2>London</h2>
```

```
<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
```

```
</article>
```

```
<article>
```

```
<h2>Paris</h2>
```

```
<p>Paris is the capital and most populous city of France.</p>
```

```
</article>
```

```
<article>
```

```
<h2>Tokyo</h2>
```

```
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.</p>
```

```
</article>
```

```
</article>
```

<div> in <article>:

```
<article>
```

```
<h2>Famous Cities</h2>
```

```
<div class="city">
```

```
<h2>London</h2>
```

```
<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
```

```
</div>
```

```
<div class="city">
```

```
<h2>Paris</h2>
```

```
<p>Paris is the capital and most populous city of France.</p>
```

```
</div>
```

```
<div class="city">
```

```
<h2>Tokyo</h2>
```

```
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.</p>
```

```
</div>
```

```
</article>
```

<div> in <section> in <article>:

```
<article>
```

```
<section>
```

## <h2>Famous Cities</h2>

<div class="city">

<h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>

</div>

<div class="city">

<h2>Paris</h2>

<p>Paris is the capital and most populous city of France.</p>

</div>

<div class="city">

<h2>Tokyo</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.</p>

</div>

</section>

<section>

<h2>Famous Countries</h2>

<div class="country">

<h2>England</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>

</div>

<div class="country">

<h2>France</h2>

<p>Paris is the capital and most populous city of France.</p>

</div>

<div class="country">

<h2>Japan</h2>

<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.</p>

</div>

</section>

</article>

# HTML5 Style Guide and Coding Conventions

## HTML Coding Conventions

Web developers are often uncertain about the coding style and syntax to use in HTML.

Between 2000 and 2010, many web developers converted from HTML to XHTML.

With XHTML, developers were forced to write valid and "well-formed" code.

HTML5 is a bit more sloppy when it comes to code validation.

## Be Smart and Future Proof

A consistent use of style, makes it easier for others to understand your HTML.

In the future, programs like XML readers, may want to read your HTML.

Using a well-formed-"close to XHTML" syntax, can be smart.

Always keep your code tidy, clean, and well-formed.

## Use Correct Document Type

Always declare the document type as the first line in your document:

```
<!DOCTYPE html>
```

If you want consistency with lower case tags, you can use:

```
<!doctype html>
```

## Use Lower Case Element Names

HTML5 allows mixing uppercase and lowercase letters in element names.

We recommend using lowercase element names because:

- Mixing uppercase and lowercase names is bad
- Developers normally use lowercase names (as in XHTML)
- Lowercase look cleaner
- Lowercase are easier to write



Bad:

```
<SECTION>
  <p>This is a paragraph.</p>
</SECTION>
```

Very Bad:

```
<Section>
  <p>This is a paragraph.</p>
</SECTION>
```

Good:

```
<section>
  <p>This is a paragraph.</p>
</section>
```

## Close All HTML Elements

In HTML5, you don't have to close all elements (for example the `<p>` element).

We recommend closing all HTML elements.

Bad:

```
<section>
  <p>This is a paragraph.
  <p>This is a paragraph.
</section>
```

Good:

```
<section>
  <p>This is a paragraph.</p>
  <p>This is a paragraph.</p>
</section>
```

## Close Empty HTML Elements

In HTML5, it is optional to close empty elements.

Allowed:

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

Also Allowed:

```
<meta charset="utf-8" />
```

However, the closing slash (/) is REQUIRED in XHTML and XML.

If you expect XML software to access your page, it is a good idea to keep the closing slash!

# Use Lower Case Attribute Names

HTML5 allows mixing uppercase and lowercase letters in attribute names.

We recommend using lowercase attribute names because:

- Mixing uppercase and lowercase names is bad
- Developers normally use lowercase names (as in XHTML)
- Lowercase look cleaner
- Lowercase are easier to write

Bad:

```
<div CLASS="menu">
```

Good:

```
<div class="menu">
```

# Quote Attribute Values

HTML5 allows attribute values without quotes.

We recommend quoting attribute values because:

- Mixing uppercase and lowercase values is bad
- Quoted values are easier to read
- You MUST use quotes if the value contains spaces

Very bad:

This will not work, because the value contains spaces:

```
<table class=table striped>
```

Bad:

```
<table class=striped>
```

Good:

```
<table class="striped">
```

# Image Attributes

Always add the "alt" attribute to images. This attribute is important when the image for some reason cannot be displayed. Also, always define image width and height. It reduces flickering because the browser can reserve space for the image before loading.

Bad:

```

```

Good:

```

```

# Spaces and Equal Signs

HTML5 allows spaces around equal signs. But space-less is easier to read, and groups entities better together.

Bad:

```
<link rel = "stylesheet" href = "styles.css">
```

Good:

```
<link rel="stylesheet" href="styles.css">
```

# Avoid Long Code Lines

When using an HTML editor, it is inconvenient to scroll right and left to read the HTML code.

Try to avoid code lines longer than 80 characters.

# Blank Lines and Indentation

Do not add blank lines without a reason.

For readability, add blank lines to separate large or logical code blocks.

For readability, add two spaces of indentation. Do not use the tab key.

Do not use unnecessary blank lines and indentation. It is not necessary to indent every element:

Unnecessary:

```
<body>

  <h1>Famous Cities</h1>

  <h2>Tokyo</h2>
```

```
<p>
  Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
  and the most populous metropolitan area in the world.
  It is the seat of the Japanese government and the Imperial Palace,
  and the home of the Japanese Imperial Family.
</p>
```

```
</body>
```

## Better:

```
<body>
```

```
<h1>Famous Cities</h1>
```

```
<h2>Tokyo</h2>
```

```
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.
It is the seat of the Japanese government and the Imperial Palace,
and the home of the Japanese Imperial Family.</p>
```

```
</body>
```

## Table Example:

```
<table>
  <tr>
    <th>Name</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>A</td>
    <td>Description of A</td>
  </tr>
  <tr>
    <td>B</td>
    <td>Description of B</td>
  </tr>
</table>
```

## List Example:

```
<ol>
  <li>London</li>
  <li>Paris</li>
  <li>Tokyo</li>
</ol>
```

## Omitting <html> and <body>?

In the HTML5 standard, the <html> tag and the <body> tag can be omitted.

The following code will validate as HTML5:

## Example

```
<!DOCTYPE html>
<head>
```

```
<title>Page Title</title>
</head>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

### **We do not recommend omitting the <html> and <body> tags.**

The <html> element is the document root. It is the recommended place for specifying the page language:

```
<!DOCTYPE html>
<html lang="en-US">
```

Declaring a language is important for accessibility applications (screen readers) and search engines.

Omitting <html> or <body> can crash DOM and XML software.

Omitting <body> can produce errors in older browsers (IE9).

## Omitting <head>?

In the HTML5 standard, the <head> tag can also be omitted.

By default, browsers will add all elements before <body>, to a default <head> element.

You can reduce the complexity of HTML, by omitting the <head> tag:

### Example

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

<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>

</html>
```

### **We do not recommend omitting the <head> tag.**

Omitting tags is unfamiliar to web developers. It needs time to be established as a guideline.

## Meta Data

The <title> element is required in HTML5. Make the title as meaningful as possible:

```
<title>HTML5 Syntax and Coding Style</title>
```

To ensure proper interpretation, and correct search engine indexing, both the language and the character encoding should be defined as early as possible in a document:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Syntax and Coding Style</title>
</head>
```

## Setting The Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

The viewport is the user's visible area of a web page. It varies with the device, and will be smaller on a mobile phone than on a computer screen.

You should include the following <meta> viewport element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

A <meta> viewport element gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

**Tip:** If you are browsing this page with a phone or a tablet, you can click on the two links below to see the difference.

## HTML Comments

Short comments should be written on one line, like this:

```
<!-- This is a comment -->
```

Comments that spans more than one line, should be written like this:

```
<!--
  This is a long comment example. This is a long comment example.
  This is a long comment example. This is a long comment example.
-->
```

Long comments are easier to observe if they are indented two spaces.

# Style Sheets

Use simple syntax for linking to style sheets (the type attribute is not necessary):

```
<link rel="stylesheet" href="styles.css">
```

Short rules can be written compressed, on one line, like this:

```
p.intro {font-family: Verdana; font-size: 16em;}
```

Long rules should be written over multiple lines:

```
body {  
  background-color: lightgrey;  
  font-family: "Arial Black", Helvetica, sans-serif;  
  font-size: 16em;  
  color: black;  
}
```

- Place the opening bracket on the same line as the selector
- Use one space before the opening bracket
- Use two spaces of indentation
- Use semicolon after each property-value pair, including the last
- Only use quotes around values if the value contains spaces
- Place the closing bracket on a new line, without leading spaces
- Avoid lines over 80 characters

## Loading JavaScript in HTML

Use simple syntax for loading external scripts (the type attribute is not necessary):

```
<script src="myscript.js">
```

## Accessing HTML Elements with JavaScript

A consequence of using "untidy" HTML styles, might result in JavaScript errors.

These two JavaScript statements will produce different results:

### Example

```
var obj = getElementById("Demo")
```

```
var obj = getElementById("demo")
```

## Use Lower Case File Names

Some web servers (Apache, Unix) are case sensitive about file names: "london.jpg" cannot be accessed as "London.jpg".

Other web servers (Microsoft, IIS) are not case sensitive: "london.jpg" can be accessed as "London.jpg" or "london.jpg".

If you use a mix of upper and lower case, you have to be extremely consistent.

If you move from a case insensitive to a case sensitive server, even small errors will break your web!

To avoid these problems, always use lower case file names.

## File Extensions

HTML files should have a **.html** or **.htm** extension.

CSS files should have a **.css** extension.

JavaScript files should have a **.js** extension.

## Differences Between .htm and .html

There is no difference between the .htm and .html extensions. Both will be treated as HTML by any web browser or web server.

The differences are cultural:

.htm "smells" of early DOS systems where the system limited the extensions to 3 characters.

.html "smells" of Unix operating systems that did not have this limitation.

## Technical Differences

When a URL does not specify a filename (like <https://www.w3schools.com/css/>), the server returns a default filename. Common default filenames are index.html, index.htm, default.html, and default.htm.

If your server is configured only with "index.html" as default filename, your file must be named "index.html", not "index.htm."

However, servers can be configured with more than one default filename, and normally you can set up as many default filenames as needed.

Anyway, the full extension for HTML files is .html, and there's no reason it should not be used.



## PRAKTIKUM

HTML Attributes

HTML Headings, Rules, & Comments

HTML Paragraphs

HTML Text Formatting

HTML Styles

HTML Links

HTML Images

HTML Tables

HTML Lists

## LATIHAN & TUGAS

Membuat Biodata Mahasiswa (data pribadi) online seperti contoh dibawah ini:

### Biodata Mahasiswa



Nama	: ISA ACHSANU
NIM	: A11.2017.10671
Dosen Wali	: Solichul Huda
Status	: Aktif
Alamat Semarang	: JL SIDODRAJ____
Alamat Asal	: JL SIDODRAJ____
No Hp	: 08995____
E-mail	: 111201710671[a]mhs.dinus.ac.id 0 isa.achsanu[a]gmail.com
Agama	: Islam

#### Jadwal KRS

No	Kode Mata Kuliah	Kelompok	Matakuliah	SKS	Status
1	A11.54101	A11.4114	KALKULUS 1	4	B
2	A11.54102	A11.4114	FISIKA 1	4	B
3	A11.54105	A11.4114	DASAR PEMROGRAMAN	4	B
4	AF201704	A11.4114	DASAR DASAR KOMPUTASI	2	B
5	N201702	A11.4114	BAHASA INGGRIS	2	B
6	N201705	A11.4114	PENDIDIKAN AGAMA	2	B
7	U201704	A11.4114	PENGANTAR TEKNOLOGI INFORMASI	2	B