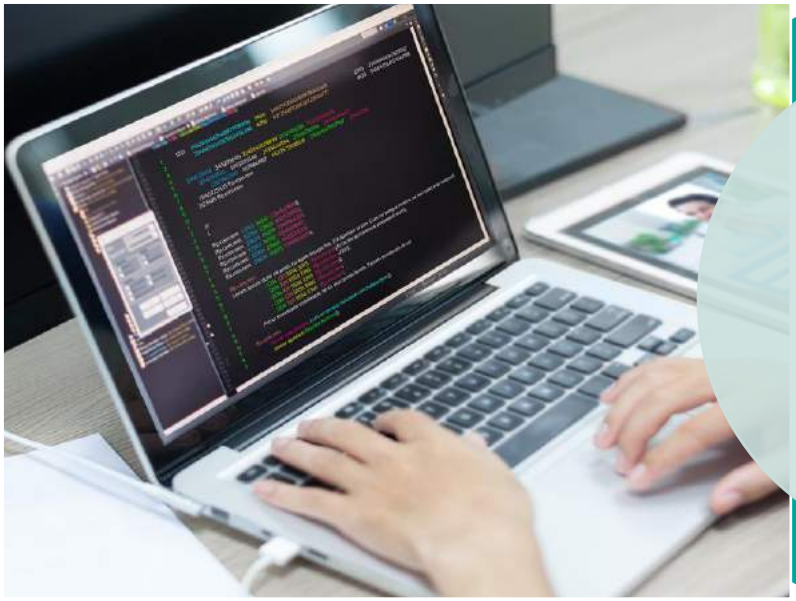


مختبر برمجة انترنت

يمثل هذا المختبر الجانب العملي لمادة برمجة الانترنت النظرية للمرحلة الرابعة، و من خلاله يتعلم الطالب أساسيات تصميم المواقع وكيفية استخدام الServer لخزن المعلومات الخاصه بالمستخدم.

اهداف المختبر

1. تعليم الطالب على كيفية تصميم موقع الكتروني باستخدام لغة HTML و الـ JavaScript وذلك باستخدام ال Text editor ومن ثم عرض النتائج على الـ Browser.
2. تدريب الطالب على تصميم موقع الكتروني وارسال المعلومات الخاصه بالموقع الى Server معين وكذلك تصميم قواعد بيانات لجميع المعلومات المسجله في الموقع وارسالها الى الـ Server ويتم كل ذلك باستخدام لغة الـ PHP و MySql وكذلك استخدام الـ Wamp Server.

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

HTML Introduction, Basic and Elements

What is HTML?

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup

Cont.

- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

A Simple HTML Document

Example

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

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

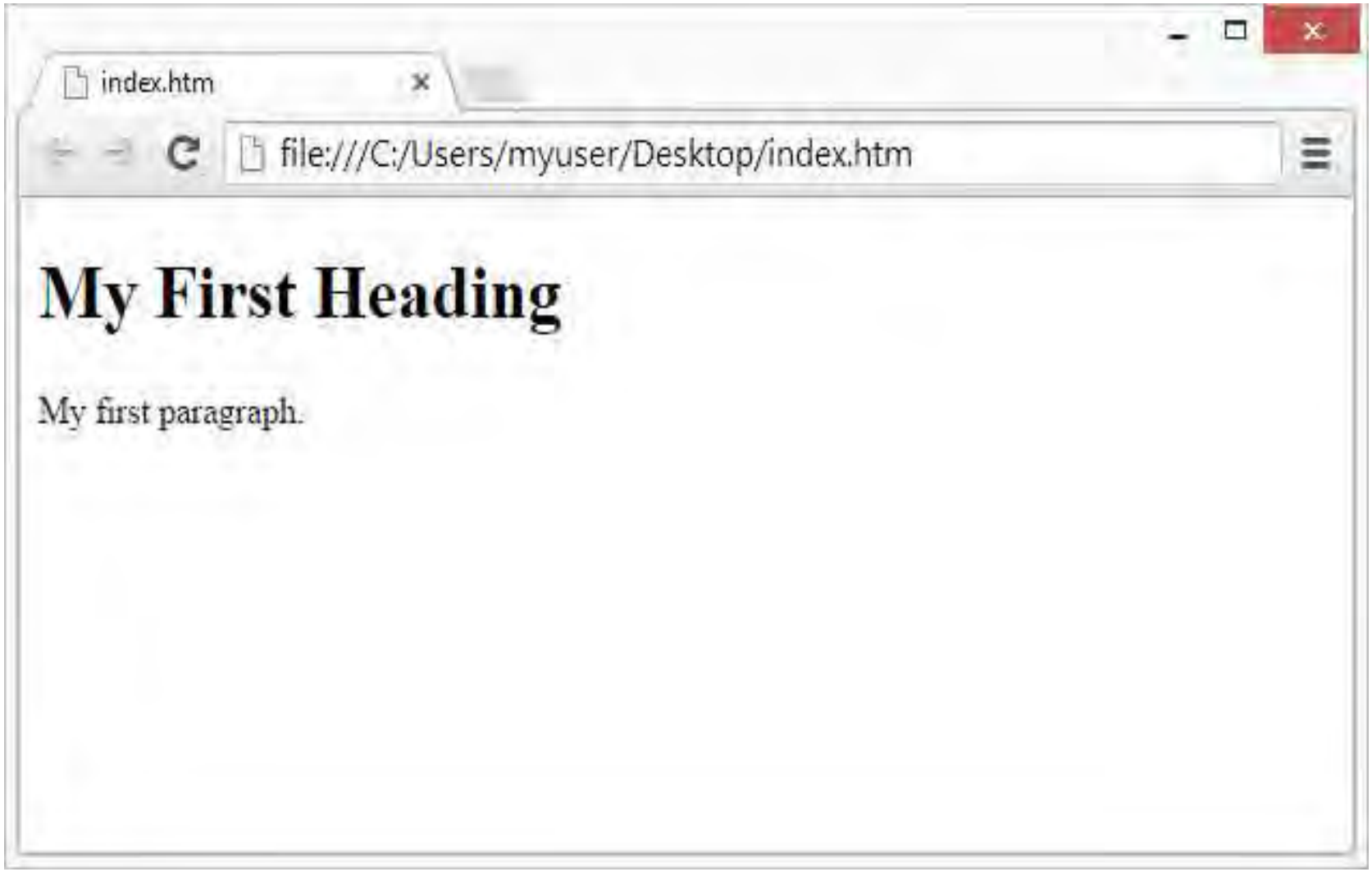
- The <!DOCTYPE html> declaration defines this document to be HTML5
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the document
- The <title> element specifies a title for the document
- The <body> element contains the visible page content
- The <h1> element defines a large heading
- The <p> element defines a paragraph

HTML Tags

- HTML tags are element names surrounded by angle brackets:
- <tagname>content goes here...</tagname>
- HTML tags normally come **in pairs** like and
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name
- **Tip:** The start tag is also called the **opening tag**, and the end tag the **closing tag**.

Web Browsers

- The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them.
- The browser does not display the HTML tags, but uses them to determine how to display the document:


index.htm
file:///C:/Users/myuser/Desktop/index.htm
My First Heading
My first paragraph.

HTML Page Structure.

- Below is a visualization of an HTML page structure:
- <html>
- <head>
- <title>Page title</title>
- </head>
- <body>
- <h1>This is a heading</h1>
- This is a paragraph.
- This is another paragraph.
- </body>
- </html>
- **Note:** Only the content inside the <body> section (the white area above) is displayed in a browser.

The <!DOCTYPE> Declaration.

- The <!DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.
- It must only appear once, at the top of the page (before any HTML tags).
- The <!DOCTYPE> declaration is not case sensitive.
- The <!DOCTYPE> declaration for HTML5 is:
- <!DOCTYPE html>

HTML Documents.

- All HTML documents must start with a document type declaration: **<!DOCTYPE html>**.
- The HTML document itself begins with **<html>** and ends with **</html>**.
- The visible part of the HTML document is between **<body>** and **</body>**.

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

HTML Heading

What is HTML Headings?

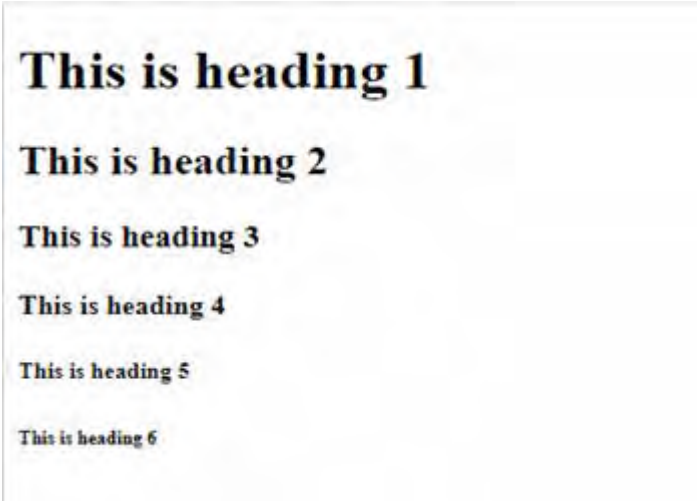
- HTML headings are defined with the **<h1>** to **<h6>** tags.
- <h1> defines the most important heading. <h6> defines the least important heading:

Cont.

Example

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
```

- The output is:



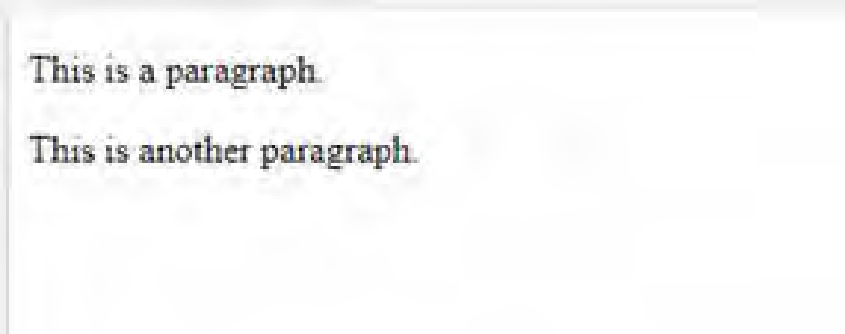
HTML Paragraphs

- HTML paragraphs are defined with the **<p>** tag:

Example

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

The output is:



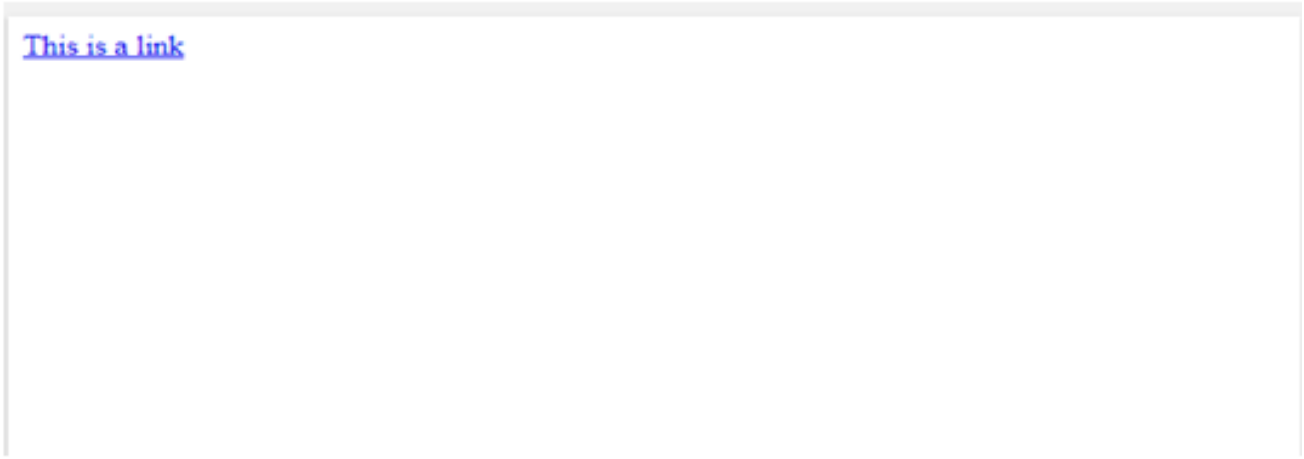
HTML Links

- HTML links are defined with the **<a>** tag:

Example

```
<a href="https://www.microsoft.com">This is a link</a>
```

The output is:



The link's destination is specified in the **href attribute**.

Attributes are used to provide additional information about HTML elements.

HTML Elements

- An HTML element usually consists of a **start** tag and **end** tag, with the content inserted in between:
- <tagname>Content goes here...</tagname>
- The HTML **element** is everything from the start tag to the end tag:
- My first paragraph.

Cont.

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	

HTML elements with no content are called empty elements. Empty elements do not have an end tag, such as the
 element (which indicates a line break).

Nested HTML Elements

- HTML elements can be nested (elements can contain elements).
- All HTML documents consist of nested HTML elements.
- This example contains four HTML elements:

Cont.

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Cont.

- **Example Explained**
- The **<html>** element defines the **whole document**.
- It has a **start** tag <html> and an **end** tag </html>.
- The element **content** is another HTML element (the <body> element).>
- The **<body>** element defines the **document body**.
- It has a **start** tag <body> and an **end** tag </body>.
- The element **content** is two other HTML elements (<h1> and <p>).

Cont.

- The **<h1>** element defines a **heading**.
- It has a **start** tag <h1> and an **end** tag </h1>.
- The element **content** is: My First Heading.
- The ** ** element defines a **paragraph**.
- It has a **start** tag and an **end** tag .
- The element **content** is: My first paragraph.

Do Not Forget the End Tag

- Some HTML elements will display correctly, even if you forget the end tag:

Example

```
<html>
<body>

<p>This is a paragraph
<p>This is a paragraph

</body>
</html>
```

The example above works in all browsers, because the closing tag is considered optional.

Never rely on this. It might produce unexpected results and/or errors if you forget the end tag.

Empty HTML Elements

- HTML elements with no content are called empty elements.
-
 is an empty element without a closing tag (the
 tag defines a line break).
- Empty elements can be "closed" in the opening tag like this:
.
- HTML5 does not require empty elements to be closed. But if you want stricter validation, or if you need to make your document readable by XML parsers, you must close all HTML elements properly.

Use Lowercase Tags

- HTML tags are not case sensitive: <P> means the same as <p>.
- The HTML5 standard does not require lowercase tags, but lowercase in HTML is **recommended**.

HTML Line Breaks

- The HTML **
** element defines a **line break**.
- Use
 if you want a line break (a new line) without starting a new paragraph:

Example

```
<p>This is<br>a paragraph<br>with line breaks.</p>
```

The
 tag is an empty tag, which means that it has no end tag.

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

HTML Attributes

Attributes provide additional information about HTML elements.

HTML Attributes

- All HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

The href Attribute

HTML links are defined with the **<a>** tag. The link address is specified in the **href** attribute:

Example

```
<a href="https://www.w3schools.com">This is a link</a>
```

The HTML <pre> Element

The HTML <pre> element defines preformatted text.

The text inside a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks:

Example

```
<pre>
  My Bonnie lies over the ocean.

  My Bonnie lies over the sea.

  My Bonnie lies over the ocean.
```

Cont.

```
  Oh, bring back my Bonnie to me.
</pre>
```

HTML Images

HTML images are defined with the **<img>** tag.

The source file (src), alternative text (alt), width, and height are provided as attributes:

Example

```
<img src="image1.jpg" alt="this is my
image1" width="104" height="142">
```

The src Attribute

HTML images are defined with the **<img>** tag.

The filename of the image source is specified in the **src** attribute:

Example

```
<img src="img_girl.jpg">
```

The width and height Attributes

Images in HTML has a set of size attributes, which specifies the width and height of the image:

Example

```
<img src="img_girl.jpg" width="500" height="600">
```

The image size is specified in pixels: width="500" means 500 pixels wide.

The alt Attribute

The **alt** attribute specifies an alternative text to be used, when an image cannot be displayed.

Example

```
<img src="img_girl.jpg" alt="Girl with a jacket">
```

The alt attribute is also useful if the image does not exist:

Example

What happens if we try to display an image that does not exist:

```
<img src="img_typo.jpg" alt="Girl with a jacket">
```

The title Attribute

Here, a **title** attribute is added to the **<p>** element. The value of the title attribute will be displayed as a tooltip when you mouse over the paragraph:

Example

```
<p title="I'm a tooltip">
This is a paragraph.
</p>
```

Single or Double Quotes?

Double quotes around attribute values are the most common in HTML, but single quotes can also be used.

In some situations, when the attribute value itself contains double quotes, it is necessary to use single quotes:

```
<p title='John "ShotGun" Nelson'>
```

Or vice versa:

```
<p title="John 'ShotGun' Nelson">
```

The HTML Style Attribute

Setting the style of an HTML element, can be done with the **style attribute**.

The HTML style attribute has the following **syntax**:

```
<tagname style="property:value;">
```

The ***property*** is a CSS property. The ***value*** is a CSS value.

The style Attribute

The style attribute is used to specify the styling of an element, like color, font, size etc.

Example

```
<p style="color:red">I am a paragraph</p>
```

HTML Background Color

The **background-color** property defines the background color for an HTML element.

This example sets the background color for a page to powderblue:

Example

```
<body style="background-color:powderblue;">

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

</body>
```

HTML Formatting Elements

In the previous chapter, you learned about the HTML **style attribute**.

HTML also defines special **elements** for defining text with a special **meaning**.

HTML uses elements like <b> and <i> for formatting output, like **bold** or *italic* text.

Formatting elements were designed to display special types of text.

Cont.

Example

```
<body>

<p>This text is normal.</p>

<p><b>This text is bold.</b> <br>
<strong> - Important text</strong><br>
<i> - Italic text </i><br>
<em> - Emphasized text</em><br>
<mark> - Marked text </mark><br>
<small> - Small text</small><br>
<del> - Deleted text </del><br>
<ins> - Inserted text</ins><br>
<u> -underlined text </u>
<sub> - Subscript text </sub>
<sup> - Superscript text </sup>
</p>

</body>
```

Cont.

The output will be :-

This text is normal.

This text is bold.
- **Important text**
- *Italic text*
- *Emphasized text*
- Marked text
- Small text
- ~~Deleted text~~
- Inserted text
-underlined text - Subscript text - Superscript text

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

HTML Comment Tags

You can add comments to your HTML source by using the following syntax:

<!-- Write your comments here -->

Notice that there is an exclamation point (!) in the opening tag, but not in the closing tag.

Note: Comments are not displayed by the browser, but they can help document your HTML source code.

With comments you can place notifications and reminders in your HTML:

Cont.

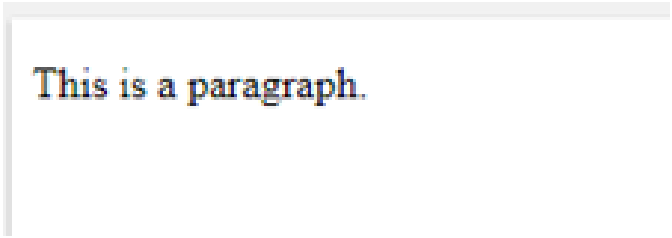
Example

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

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

<!-- Remember to add more information here -->
```

output»



Cont.

Comments are also great for debugging HTML, because you can comment out HTML lines of code, one at a time, to search for errors:

Example

```
<!-- Do not display this at the moment
<img border="0" src="pic_mountain.jpg" alt="Mountain">
-->
```

- ***HTML Tables and List***

Defining an HTML Table

An HTML table is defined with the **<table>** tag.

Each table row is defined with the **<tr>** tag. A table header is defined with the **<th>** tag. By default, table headings are bold and centered. A table data/cell is defined with the **<td>** tag.

Example

```
<table style="width:100%">
  <tr>
    <th>Firstnamek</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Note: The <td> elements are the data containers of the table.
They can contain all sorts of HTML elements; text, images, lists, other tables, etc.

The output:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

HTML Table - Adding a Border

If you do not specify a border for the table, it will be displayed without borders.

A border is set using the CSS **border** property:

Example

```
table, th, td {
    border: 1px solid black;
}
```

Remember to define borders for both the table and the table cells.

The output:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

HTML Table - Collapsed Borders

If you want the borders to collapse into one border, add the CSS **border-collapse** property:

Example

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
```

the output:-

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

HTML Table - Adding Cell Padding

Cell padding specifies the space between the cell content and its borders.
If you do not specify a padding, the table cells will be displayed without padding.
To set the padding, use the CSS **padding** property:

Example

```
th, td {
    padding: 15px;
}
```

the output:-

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

HTML Table - Left-align Headings

By default, table headings are bold and centered.

To left-align the table headings, use the CSS **text-align** property:

Example

```
th {
    text-align: left;
}
```

the output:-

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

HTML Table - Adding Border Spacing

Border spacing specifies the space between the cells.

To set the border spacing for a table, use the CSS **border-spacing** property:

Example

```
table {
    border-spacing: 5px;
}
```

Note: If the table has collapsed borders, border-spacing has no effect.

the output:-

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

HTML Table - Cells that Span Many Columns

To make a cell span more than one column, use the **colspan** attribute:

Example

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
     <td>Bill Gates</td>
    <td>55577854</td>
     <td>55577855</td>
  </tr>
</table>
```

the output:-

Name	Telephone	
Bill Gates	55577854	55577855

HTML Table - Cells that Span Many Rows

To make a cell span more than one row, use the **rowspan** attribute:

Example

```
<table style="width:100%">
  <tr>
    <th>Name:</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>55577854</td>
  </tr>
  <tr>
    <td>55577855</td>
  </tr>
</table>
```

the output:-

Name:	Bill Gates
Telephone:	55577854
	55577855

HTML Table - Adding a Caption

To add a caption to a table, use the **<caption>** tag:

Example

```
<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

Note: The <caption> tag must be inserted immediately after the <table> tag.

the output:-

Monthly savings

Month	Savings
January	$100
February	$50

A Special Style for One Table

To define a special style for a special table, add an **id** attribute to the table:

Example

```
<table id="t01">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Now you can define a special style for this table:

```
table#t01 {
    width: 100%;

    background-color: #f1f1c1;
}
```

And add more styles:

```
table#t01 tr:nth-child(even) {
    background-color: red;
}
table#t01 tr:nth-child(odd) {
    background-color: white;
}
table#t01 th {
    color: white;
    background-color: black;
}
```

the output:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Unordered HTML List

An unordered list starts with the **<ul>** tag. Each list item starts with the **<li>** tag.

The list items will be marked with bullets (small black circles) by default:

Example

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Unordered HTML List - Choose List Item Marker

The CSS **list-style-type** property is used to define the style of the list item marker:

Value	Description
disc	Sets the list item marker to a bullet (default)
circle	Sets the list item marker to a circle
square	Sets the list item marker to a square
none	The list items will not be marked

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

Ordered HTML List

An ordered list starts with the **<ol>** tag. Each list item starts with the **<li>** tag.

The list items will be marked with numbers by default:

Example

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Ordered HTML List - The Type Attribute

The **type** attribute of the <ol> tag, defines the type of the list item marker:

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

Numbers:

```
<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Uppercase Letters:

```
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Lowercase Letters:

```
<ol type="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Uppercase Roman Numbers:

```
<ol type="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Lowercase Roman Numbers:

```
<ol type="i">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

HTML Description Lists

HTML also supports description lists.

A description list is a list of terms, with a description of each term.

The **<dl>** tag defines the description list, the **<dt>** tag defines the term (name), and the **<dd>** tag describes each term:

Example

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

The output:

Coffee
- black hot drink
Milk
- white cold drink

Nested HTML Lists

List can be nested (lists inside lists):

Example

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Note: List items can contain new list, and other HTML elements, like images and links, etc.

The output:

- Coffee
- Tea
  - Black tea
  - Green tea
- Milk

HTML Block and Inline Elements

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The element is a block-level element.

Examples of block-level elements:

-
- <h1> - <h6>
- <p>
- <form>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline element inside a paragraph.

Examples of inline elements:

-
- <a>
- <img>

The Element

The element is often used as a container for other HTML elements.
The element has no required attributes, but both **style** and **class** are common.
When used together with CSS, the element can be used to style blocks of content:

Example

```
<div style="background-color:black;color:white;padding:20px;">
  <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>
```

The Element

The element is often used as a container for some text.
The element has no required attributes, but both **style** and **class** are common.
When used together with CSS, the element can be used to style parts of the text:

Example

```
<h1>My <span style="color:red">Important</span> Heading</h1>
```

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

HTML Form

The <form> Element

The HTML **<form>** element defines a form that is used to collect user input:

```
<form>
.
form elements
.
</form>
```

The <input> Element

The **<input>** element is the most important form element.
The <input> element can be displayed in several ways, depending on the **type** attribute.
Here are some examples:

Type	Description
<input type="text">	Defines a one-line text input field
<input type="radio">	Defines a radio button (for selecting one of many choices)
<input type="submit">	Defines a submit button (for submitting the form)

Text Input

<input type="text"> defines a one-line input field for **text input**:

Example

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

This is how it will look like in a browser:

First name:

Last name:

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters.

Radio Button Input

<input type="radio"> defines a **radio button**.
Radio buttons let a user select ONE of a limited number of choices:

Example

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

This is how the HTML code above will be displayed in a browser:

- (•) Male
- ( ) Female
- ( ) Other

The Submit Button

<input type="submit"> defines a button for **submitting** the form data to a **form-handler**.
The form-handler is typically a server page with a script for processing input data.
The form-handler is specified in the form's **action** attribute:

Example

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

This is how the HTML code above will be displayed in a browser:

First name:

Mickey

Last name:

Mouse

Submit

If you omit the submit button's value attribute, the button will get a default text "Submit"

The <select> Element

The **<select>** element defines a **drop-down list**:

Example

```
<select name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

Output:-

The **<option>** elements defines an option that can be selected.
By default, the first item in the drop-down list is selected.
To define a pre-selected option, add the **selected** attribute to the option:

Example

```
<option value="fiat" selected>Fiat</option>
```

Visible Values:

Use the **size** attribute to specify the number of visible values:

Example

```
<select name="cars" size="3">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

Output:

Submit

Allow Multiple Selections:

Use the **multiple** attribute to allow the user to select more than one value:

```
<select name="cars" size="4" multiple>
```

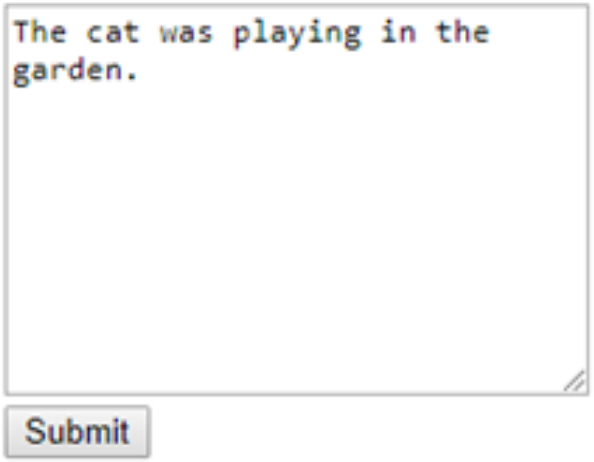
The <textarea> Element

The **<textarea>** element defines a multi-line input field (**a text area**):

Example

```
<textarea name="message" rows="10" cols="30">
The cat was playing in the garden.
</textarea>
```

The **rows** attribute specifies the visible number of lines in a text area.
The **cols** attribute specifies the visible width of a text area.
This is how the HTML code above will be displayed in a browser:



The <button> Element

The **<button>** element defines a clickable **button**:

Example

```
<button type="button" onclick="alert('Hello World!')">Click
Me!</button>
```

This is how the HTML code above will be displayed in a browser:

Input Type Password

<input type="password"> defines a **password field**:

The characters in a password field are masked (shown as asterisks or circles).

Input Type Reset

<input type="reset"> defines a **reset button** that will reset all form values to their default values:

```
<input type="reset">
```

If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.

Input Type Checkbox

<input type="checkbox"> defines a **checkbox**.
Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example

```
<form>
  <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
  <input type="checkbox" name="vehicle2" value="Car"> I have a car
</form>
```

This is how the HTML code above will be displayed in a browser:

- ☐ **I have a bike**
- ☐ **I have a car**

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

HTML5 Input Types

HTML5 Input Types

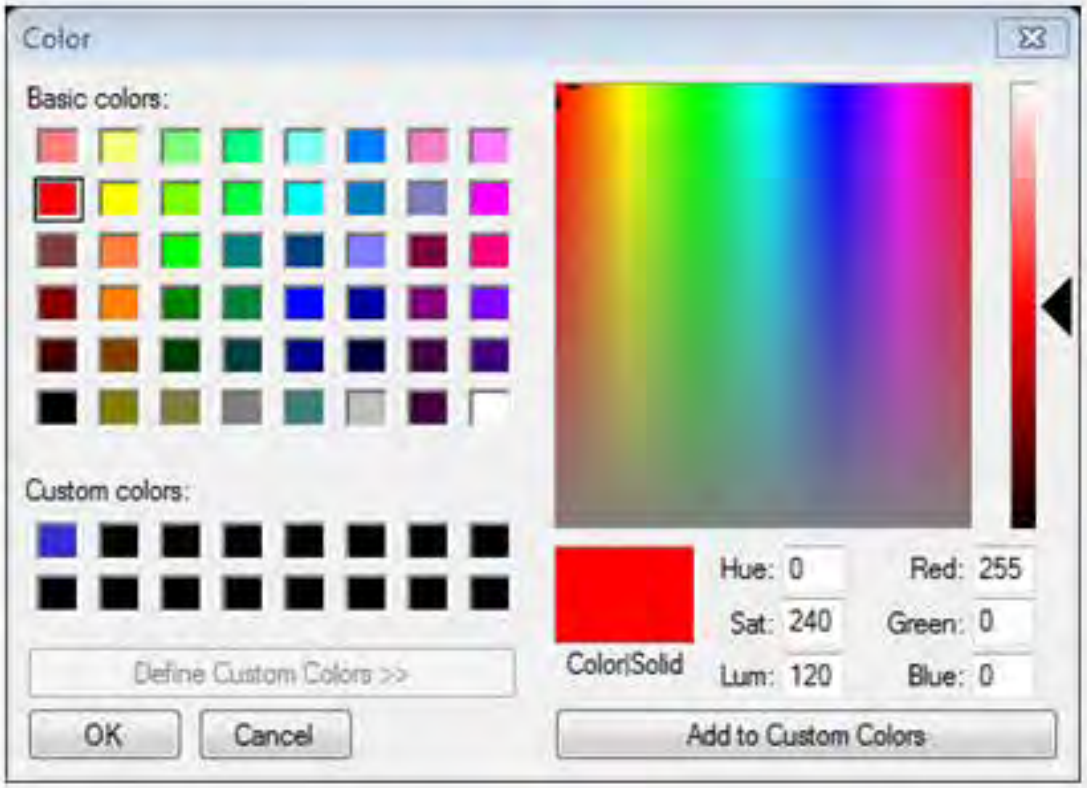
HTML5 added several new input types:

- color
- date
- email
- number
- range
- time
- week

New input types that are not supported by older web browsers, will behave as <input type="text">.

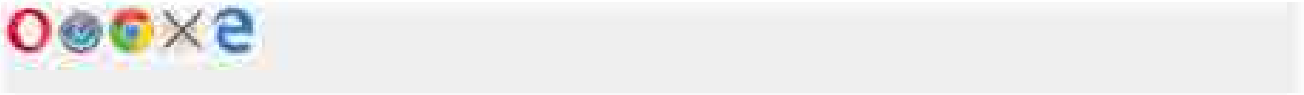
Input Type Color

The **<input type="color">** is used for input fields that should contain a color.
Depending on browser support, a color picker can show up in the input field.



Input Type Date

The **<input type="date">** is used for input fields that should contain a date.
Depending on browser support, a date picker can show up in the input field.



You can also add restrictions to dates:

Example

```
<form>
  Enter a date before 1980-01-01:
  <input type="date" name="bday" max="1979-12-31"><br>
  Enter a date after 2000-01-01:
  <input type="date" name="bday" min="2000-01-02"><br>
</form>
```

Input Type Email

The **<input type="email">** is used for input fields that should contain an e-mail address.
Depending on browser support, the e-mail address can be automatically validated when submitted.
Some smartphones recognize the email type, and adds ".com" to the keyboard to match email input.

Input Type Number

The **<input type="number">** defines a **numeric** input field.

You can also set restrictions on what numbers are accepted.

The following example displays a numeric input field, where you can enter a value from 1 to 5:

Example

```
<form>
  Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1" max="5">
</form>
```

Input Type Range

The **<input type="range">** defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the min, max, and step attributes:

Example

```
<form>
Points:  <input type="range" name="points" min="0" max="10">

<input type="submit">
</form>
```

Input Type Time

The **<input type="time">** allows the user to select a time (no time zone).

Depending on browser support, a time picker can show up in the input field.

Example

```
<form>
  Select a time:
  <input type="time" name="usr_time">

  <input type="submit">
</form>
```

Select a time: --:-- Submit

Input Type Week

The **<input type="week">** allows the user to select a week and year.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  Select a week:
  <input type="week" name="week_year">
</form>
```

Select a week: Week 47, 2017 Submit

Grouping Form Data with <fieldset>

The **<fieldset>** element is used to group related data in a form.

The **<legend>** element defines a caption for the <fieldset> element.

Example

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey"><br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

This is how the HTML code above will be displayed in a browser:

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

JAVA script

JavaScript is the programming language of HTML and the Web.

JavaScript is easy to learn.

Why Study JavaScript?

JavaScript is one of the **3 languages** all web developers **must** learn:

1. **HTML** to define the content of web pages
2. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behavior of web pages

JavaScript Can Change HTML Content

One of many JavaScript HTML methods is **getElementById()**.

This example uses the method to "find" an HTML element (with id="demo") and changes the element content (**innerHTML**) to "Hello JavaScript":

Example

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

note: JavaScript accepts both double and single quotes:

Example

```
document.getElementById('demo').innerHTML = 'Hello JavaScript';
```

JavaScript Can Change HTML Attributes

This example changes an HTML image by changing the src (source) attribute of an <img> tag:

```
document.getElementById('myImage').src='pic_bulbon.gif';
```

JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute:

Example

```
document.getElementById("demo").style.fontSize = "35px";
```

or

```
document.getElementById('demo').style.fontSize = '35px';
```

JavaScript Can Hide HTML Elements

Hiding HTML elements can be done by changing the display style:

Example

```
document.getElementById("demo").style.display = "none";
```

or

```
document.getElementById('demo').style.display = 'none';
```

JavaScript Can Show HTML Elements

Showing hidden HTML elements can also be done by changing the display style:

Example

```
document.getElementById("demo").style.display = "block";
```

or

```
document.getElementById('demo').style.display = 'block';
```

JavaScript Where To

The <script> Tag

In HTML, JavaScript code must be inserted between <script> and </script> tags.

Example

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

Old JavaScript examples may use a type attribute: <script type="text/javascript">.
The type attribute is not required. JavaScript is the default scripting language in HTML.

JavaScript Functions and Events

A JavaScript **function** is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

JavaScript in <head> or <body>

You can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

JavaScript in <head>

Example

```
<!DOCTYPE html>
<html>

<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>

<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

JavaScript in <body>

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

Placing scripts at the bottom of the <body> element improves the display speed, because script compilation slows down the display.

External JavaScript

Scripts can also be placed in external files:

External file: myScript.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension **.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

Example

```
<!DOCTYPE html>
<html>
<body>


<script src="myScript.js"></script>


</body>
</html>
```

You can place an external script reference in <head> or <body> as you like.

The script will behave as if it was located exactly where the <script> tag is located.

External scripts cannot contain <script> tags.

External JavaScript Advantages

Placing scripts in external files has some advantages:

- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page loads

To add several script files to one page - use several script tags:

Example

```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

External References

External scripts can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a script:

Example

```
<script src="https://www.w3schools.com/js/myScript1.js"></script>
```

This example uses a script located in a specified folder on the current web site:

Example

```
<script src="/js/myScript1.js"></script>
```

Thanks…

Web Programming Lab

Assistant Lecturer Saja Dh. Khudhur

JAVA Functions and Scope

Function Declarations

syntax:

```
function functionName(parameters) {
  code to be executed
}
```

Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are invoked (called upon).

Example

```
function myFunction(a, b) {
    return a * b;
}
```

Function Expressions

A JavaScript function can also be defined using an **expression**.

A function expression can be stored in a variable:

Example

```
var x = function (a, b) {return a * b};
```

After a function expression has been stored in a variable, the variable can be used as a function:

Example

```
var x = function (a, b) {return a * b};
var z = x(4, 3);
```

The function above is actually an **anonymous function** (a function without a name).

Functions Can Be Used as Values

JavaScript functions can be used as values:

Example

```
function myFunction(a, b) {
    return a * b;
}

var x = myFunction(4, 3);
```

JavaScript functions can be used in expressions:

Example

```
function myFunction(a, b) {
    return a * b;
}

var x = myFunction(4, 3) * 2;
```

JavaScript return Statement

The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>
<body>
<script>
document.write(product(5,3));
</script>
</body>
</html>
```

JavaScript Function Scope

In JavaScript there are two types of scope:

- Local scope
- Global scope

JavaScript has function scope: Each function creates a new scope.

Scope determines the accessibility (visibility) of these variables.

Variables defined inside a function are not accessible (visible) from outside the function.

Local JavaScript Variables

Variables declared within a JavaScript function, become **LOCAL** to the function.

Local variables have **local scope**: They can only be accessed within the function.

Example

```
// code here cannot use carName

function myFunction() {
    var carName = "Volvo";

    // code here can use carName

}
```

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts, and deleted when the function is completed.

Global JavaScript Variables

A variable declared outside a function, becomes **GLOBAL**.

A global variable has **global scope**: All scripts and functions on a web page can access it.

Example

```
var carName = " Volvo";

// code here can use carName

function myFunction() {

    // code here can use carName

}
```

JavaScript Variables

In JavaScript, objects and functions are also variables.

Scope determines the accessibility of variables, objects, and functions from different parts of the code.

Automatically Global

If you assign a value to a variable that has not been declared, it will automatically become a **GLOBAL** variable.

This code example will declare a global variable **carName**, even if the value is assigned inside a function.

Example

```
myFunction();

// code here can use carName

function myFunction() {
    carName = "Volvo";
}
```

Thanks…