HTML Introduction

What Is HTML?

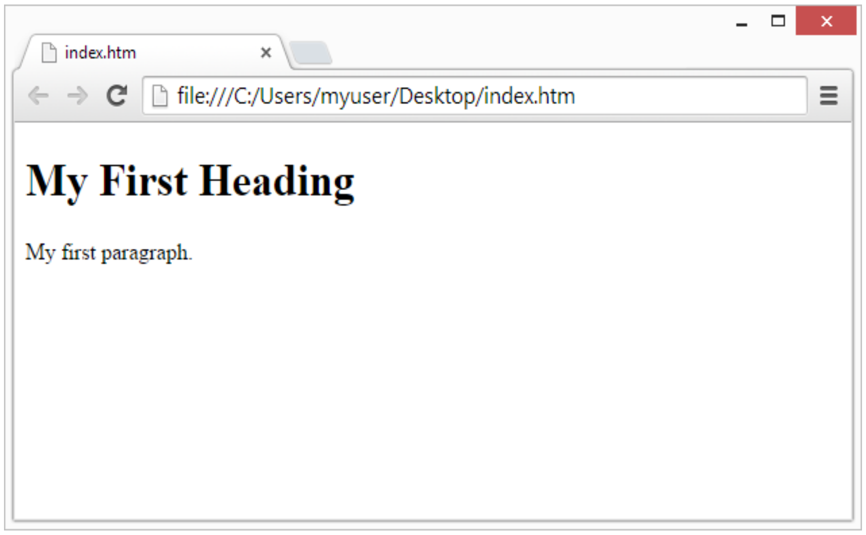
HTML Is The Standard Markup Language For Creating Web Pages.

- **HTML Stands For Hyper Text Markup Language**
- **HTML Describes The Structure Of A Web Page**
- **HTML Consists Of A Series Of Elements**
- **HTML Elements Tell The Browser How To Display The Content**
- **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**

Web Browsers

The Purpose Of A Web Browser (Chrome, Edge, 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:



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>

<P>This Is A Paragraph.</P>

<P>This Is Another Paragraph.</P>

</Body>

</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>.

Example

```
<!Doctype HTML>
<HTML>
<Head>
<Title>Title Of The Document</Title>
</Head>
<Body>
The Content Of The Document......
</Body>
</HTML>
```

The <!Doctype> Declaration Must Be The Very First Thing In Your HTML Document, Before The <HTML> Tag.

The <!Doctype> Declaration Is Not An HTML Tag; It Is An Instruction To The Web Browser About What Version Of HTML The Page Is Written In.

In HTML 4.01, The <!Doctype> Declaration Refers To A Dtd, Because HTML 4.01 Was Based On Sgml. The Dtd Specifies The Rules For The Markup Language, So That The Browsers Render The Content Correctly.

HTML5 Is Not Based On Sgml, And Therefore Does Not Require A Reference To A Dtd.

HTML Versions

Since The Early Days Of The Web, There Have Been Many Versions Of HTML

Version	Year
HTML	**1991**
HTML 2.0	**1995**
HTML 3.2	**1997**
HTML 4.01	**1999**
XHTML	**2000**
HTML5	**2014**

HTML Editors

Write HTML Using Notepad Or Textedit.Web Pages Can Be Created And Modified By Using Professional HTML Editors. However, For Learning HTML We Recommend A Simple Text Editor Like Notepad (Pc) Or Textedit (Mac).We Believe Using A Simple Text Editor Is A Good Way To Learn HTML.

Follow The Steps Below To Create Your First Web Page With Notepad Or Textedit.

Step 1: Open Notepad (Pc)

Windows 8 Or Later:

Open The Start Screen (The Window Symbol At The Bottom Left On Your Screen). Type Notepad.

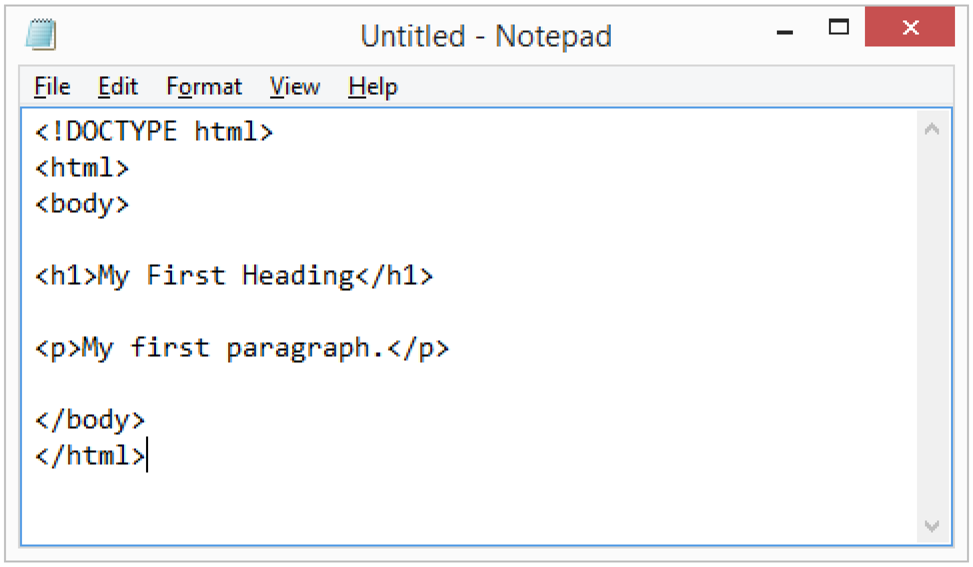
Windows 7 Or Earlier:

Open Start > Programs > Accessories > Notepad

Step 2: Write Some HTML

Write Or Copy Some HTML Into Notepad.

```
<!Doctype HTML>
<HTML>
<Body>

<H1>My First Heading</H1>

<P>My First Paragraph.</P>

</Body>
</HTML>
```

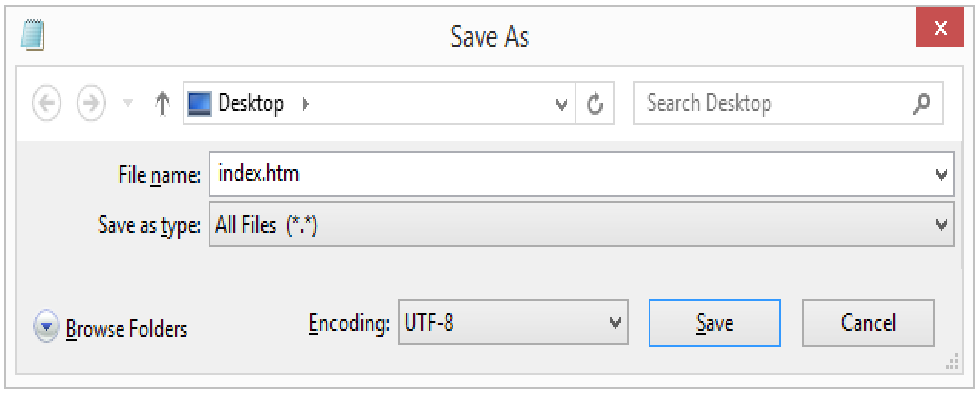


```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Step 3: Save The HTML Page

Save The File On Your Computer. Select File > Save As In The Notepad Menu.

Name The File "Index.Htm" And Set The Encoding To Utf-8 (Which Is The Preferred Encoding For HTML Files).

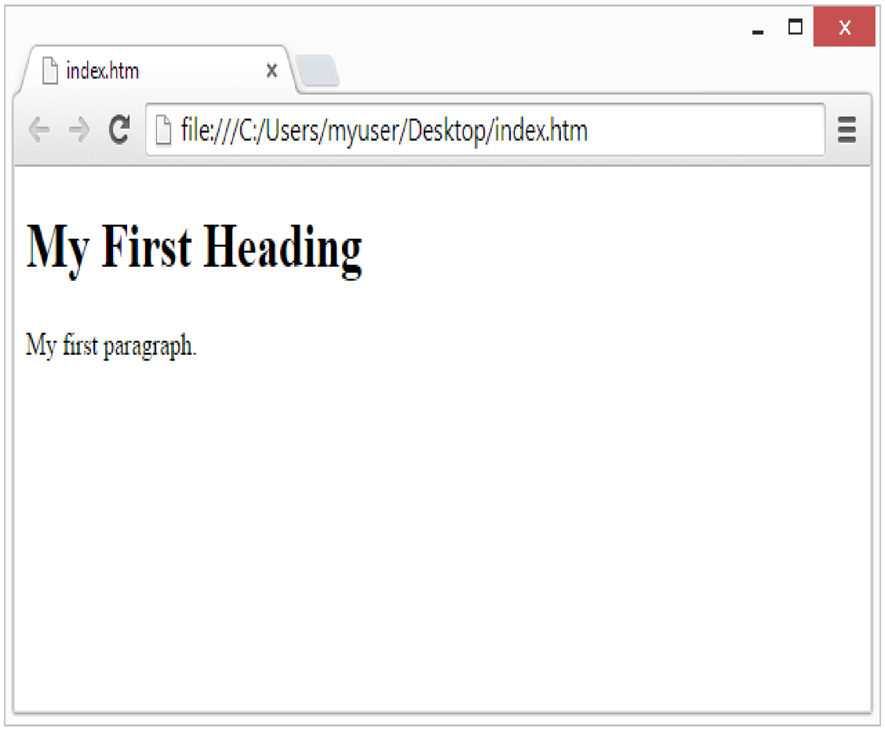


File name: index.htm
Save as type: All Files (*.*)
Encoding: UTF-8

Tep 4: View The HTML Page In Your Browser

Open The Saved HTML File In Your Favorite Browser (Double Click On The File, Or Right-Click - And Choose "Open With").

The Result Will Look Much Like This:



My First Heading

My first paragraph.

HTML Attributes

Attributes Provide Additional Information About HTML Elements.

- **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 (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 Style Attribute

The Style Attribute Is Used To Specify The Styling Of An Element, Like Color, Font, Size Etc.

Example

<P Style="Color:Red">This Is A Paragraph.</P>

Example

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

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

</body>
```

Example

نوع الخط

```
<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>
```

Example

```
<h1 style="font-size:300%;">This is a heading</h1>
<p style="font-size:160%;">This is a paragraph.</p>
```

Example

محاذاه النص

```
<h1 style="text-align:center;">Centered Heading</h1>
<p style="text-align:center;">Centered paragraph.</p>
```

Example

اطار

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Color Values قيم الالوان

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

Example

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>
<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

RGB Value

In HTML, a color can be specified as an RGB value, using this formula:

rgb(*red, green, blue*)

HEX Value

In HTML, a color can be specified using a hexadecimal value in the form:

#rrggbb

HSL Value

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(*hue, saturation, lightness*)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white

RGBA Value

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with:

rgba(*red, green, blue, alpha*)

The alpha parameter is a number between 0.0 (fully transparentشفافيه) and 1.0 (not transparent at all):

HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

hsla(*hue, saturation, lightness, alpha*)

HTML Elements

An HTML Element Usually Consists Of A Start Tag And An 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:

<P>My First Paragraph.</P>

Start Tag	Element Content	End Tag
<H1>	My First Heading	</H1>
<P>	My First Paragraph.	</P>
<Br>		

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:

Example

```
<!Doctype HTML>
<HTML>
<Body>

<H1>My First Heading</H1>
<P>My First Paragraph.</P>

</Body>
</HTML>
```

Empty HTML Elements

HTML Elements With No Content Are Called Empty Elements.

<Br> Is An Empty Element Without A Closing Tag (The <Br> Tag Defines A Line Break):

Example

```
<P>This Is A <Br> Paragraph With A Line Break.</P>
```

HTML Headings

HTML Headings Are Defined With The <H1> To <H6> Tags.

<H1> Defines The Most Important Heading. <H6> Defines The Least Important Heading:

Example

```
<H1>This Is Heading 1</H1>
<H2>This Is Heading 2</H2>
<H3>This Is Heading 3</H3>
```

HTML Links - Hyperlinks

HTML links are hyperlinks.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand.

HTML Links - Syntax

Hyperlinks are defined with the HTML <a> tag:

<a href="*url*">*link text*</a>

Example

<a href="https://www.w3schools.com/html/">Visit our HTML tutorial</a>

Definition And Usage

The <A> Tag Defines A Hyperlink, Which Is Used To Link From One Page To Another.

The Most Important Attribute Of The <A> Element Is The Href Attribute, Which Indicates The Link's Destination.

By Default, Links Will Appear As Follows In All Browsers:

- **An Unvisited Link Is Underlined And Blue**
- **A Visited Link Is Underlined And Purple**
- **An Active Link Is Underlined And Red**

HTML Links - The target Attribute

The target attribute specifies where to open the linked document.

The target attribute can have one of the following values:

- **_blank - Opens the linked document in a new window or tab**
- **_self - Opens the linked document in the same window/tab as it was clicked (this is default)**
- **Example**
- **<a href="https://www.w3schools.com/" target="_blank">Visit W3Schools!</a>**

HTML Links - Image as Link

It is common to use images as links:

Example

```
<a href="default.asp">
  <img src="smiley.gif" alt="HTML
tutorial" style="width:42px;height:42px;border:0;">
</a>
```

HTML Base

Example

Specify A Default Url And A Default Target For All Links On A Page:

```
<Head>
  <Base Href="Https://Www.W3schools.Com/" Target="_Blank">
</Head>

<Body>
<Img Src="Images/Stickman.Gif" Width="24" Height="39" Alt="Stickman">
<A Href="Tags/Tag_Base.Asp">HTML Base Tag</A>
</Body>
```

Definition And Usage

The <Base> Tag Specifies The Base Url/Target For All Relative Urls In A Document.

There Can Be At Maximum One <Base> Element In A Document, And It Must Be Inside The <Head> Element.

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:

- **<b> - Bold text**
- **<strong> - Important text**

- **<i> - Italic text**
- **<em> - Emphasized text**
- **<small> - Small text**
- **<del> - Deleted text**
- **<ins> - Inserted text**
- **<sub> - Subscript text**
- **<sup> - Superscript text**

Note: Browsers display <strong> as <b>, and <em> as <i>. However, there is a difference in the meaning of these tags: <b> and <i> defines bold and italic text, but <strong> and <em> means that the text is "important".

HTML Paragraphs Are Defined With The <P> Tag:

Example

```
<P>This Is A Paragraph.</P>
<P>This Is Another Paragraph.</P>
```

Example

```
<p>This is <sup>superscripted</sup> text.</p>
```

Example

```
<p>My favorite color is <del>blue</del> red.</p>
```

Example

```
<h2>HTML <small>Small</small> Formatting</h2>
```

Multimedia Element

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="W3schools.Jpg" Alt="W3schools.Com" Width="104" Height="142">
```

HTML <Audio> Tag

Example

Play A Sound:

```
<Audio Controls>
  <Source Src="Horse.Ogg" Type="Audio/Ogg">
  <Source Src="Horse.Mp3" Type="Audio/Mpeg">
  Your Browser Does Not Support The Audio Tag.
</Audio>
```

The <Audio> Tag Defines Sound, Such As Music Or Other Audio Streams.

Currently, There Are 3 Supported File Formats For The <Audio> Element: Mp3, Wav, And Ogg:

Browser	Mp3	Wav	Ogg
Internet Explorer	Yes	No	No
Chrome	Yes	Yes	Yes
Firefox	Yes	Yes	Yes
Safari	Yes	Yes	No
Opera	Yes	Yes	Yes

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

Example - Disc

```
<ul style="list-style-type:disc;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

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

Example

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

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>
```

Horizontal List with CSS

HTML lists can be styled in many different ways with CSS.

One popular way is to style a list horizontally, to create a navigation menu:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.gg li{
```

استخدمنا class اسمه gg يوضح ب css

```
 display:inline;

 background-color:red;

padding:16px;

}



.gg a:hover{

  color:blue;


}</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

</body>
</html>
```

HTML Forms

The <form> Element

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

<form>
.
form elements

.
</form>

An HTML form contains form elements.

Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

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>
```

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>
```

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>
```

Input Type Button

`<input type="button">` **defines a button:**

Example

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

The Submit Button

يوضح بphp

`<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>
```

The Method Attribute

The method attribute specifies the HTTP method (GET or POST) to be used when submitting the form data:

Example

```
<form action="/action_page.php" method="get">
```

When to Use GET?

The default method when submitting form data is GET.

However, when GET is used, the submitted form data will be visible in the page address field:

/action_page.php?firstname=Mickey&lastname=Mouse

Notes on GET:

- **Appends form-data into the URL in name/value pairs**
- **The length of a URL is limited (2048 characters)**
- **Never use GET to send sensitive data! (will be visible in the URL)**
- **Useful for form submissions where a user wants to bookmark the result**
- **GET is better for non-secure data, like query strings in Google**

When to Use POST?

Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.

Notes on POST:

- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked
- The Name Attribute
- Each input field must have a name attribute to be submitted.
- If the name attribute is omitted, the data of that input field will not be sent at all.
- This example will only submit the "Last name" input field:
- Example

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

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>
```

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>
```

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>
```

Allow Multiple Selections:

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

Example

```
<select name="cars" size="4" multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

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>
```

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>Firstname</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>
```

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;
}
```

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;
}
```

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;
}
```

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>
```

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>
```

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>
```

HTML Comment

HTML <!--...--> Tag

Example

An HTML Comment:

```
<!--This Is A Comment. Comments Are Not Displayed In The Browser-->

<P>This Is A Paragraph.</P>
```

The Comment Tag Is Used To Insert Comments In The Source Code. Comments Are Not Displayed In The Browsers.

You Can Use Comments To Explain Your Code, Which Can Help You When You Edit The Source Code At A Later Date. This Is Especially Useful If You Have A Lot Of Code.

Styling HTML with CSS

CSS stands for Cascading Style Sheets.

CSS describes how HTML elements are to be displayed on screen, paper, or in other media.

CSS saves a lot of work. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

- **Inline - by using the style attribute in HTML elements**
- **Internal - by using a <style> element in the <head> section**

- **External - by using an external CSS file**

Inline CSS

- **An inline CSS is used to apply a unique style to a single HTML element.**
- **An inline CSS uses the style attribute of an HTML element.**
- **This example sets the text color of the <h1> element to blue:**
- **Example**
- **<h1 style="color:blue;">This is a Blue Heading</h1>**

Internal CSS

- **An internal CSS is used to define a style for a single HTML page.**
- **An internal CSS is defined in the <head> section of an HTML page, within a <style> element:**

Example

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


<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>


<body>

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

</body>
</html>
```

External CSS

- An external style sheet is used to define the style for many HTML pages.
- With an external style sheet, you can change the look of an entire web site, by changing one file!
- To use an external style sheet, add a link to it in the <head> section of the HTML page.

Example

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>



<body>

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

</body>
</html>
```

HTML Javascript

The HTML <script> Tag

The <script> tag is used to define a client-side script (JavaScript).

The <script> element either contains script statements, or it points to an external script file through the src attribute.

Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.

To select an HTML element, JavaScript most often uses the document.getElementById() method.

This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":

Example

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

A Taste of JavaScript

Here are some examples of what JavaScript can do:

JavaScript can change HTML content

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

JavaScript can change HTML styles

```
document.getElementById("demo").style.fontSize = "25px";
document.getElementById("demo").style.color = "red";
document.getElementById("demo").style.backgroundColor = "yellow";
```

JavaScript can change HTML attributes

```
document.getElementById("image").src = "picture.gif";
```

++

INTRODUTION

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.[This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

Styling HTML with CSS

CSS saves a lot of work. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

- **Inline - by using the style attribute in HTML elements**
- **Internal - by using a <style> element in the <head> section**
- **External - by using an external CSS file**

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the style attribute of an HTML element.

This example sets the text color of the <h1> element to blue:

Example

```
<h1 style="color:blue;">This is a Blue Heading</h1>
```

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the `<head>` section of an HTML page, within a `<style>` element:

Example

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


<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>


</head>
<body>

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

</body>
</html>
```

External CSS

- **An external style sheet is used to define the style for many HTML pages.**
- **With an external style sheet, you can change the look of an entire web site, by changing one file!**

- To use an external style sheet, add a link to it in the `<head>` section of the HTML page.

Example

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>


<body>

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

</body>
</html>
```

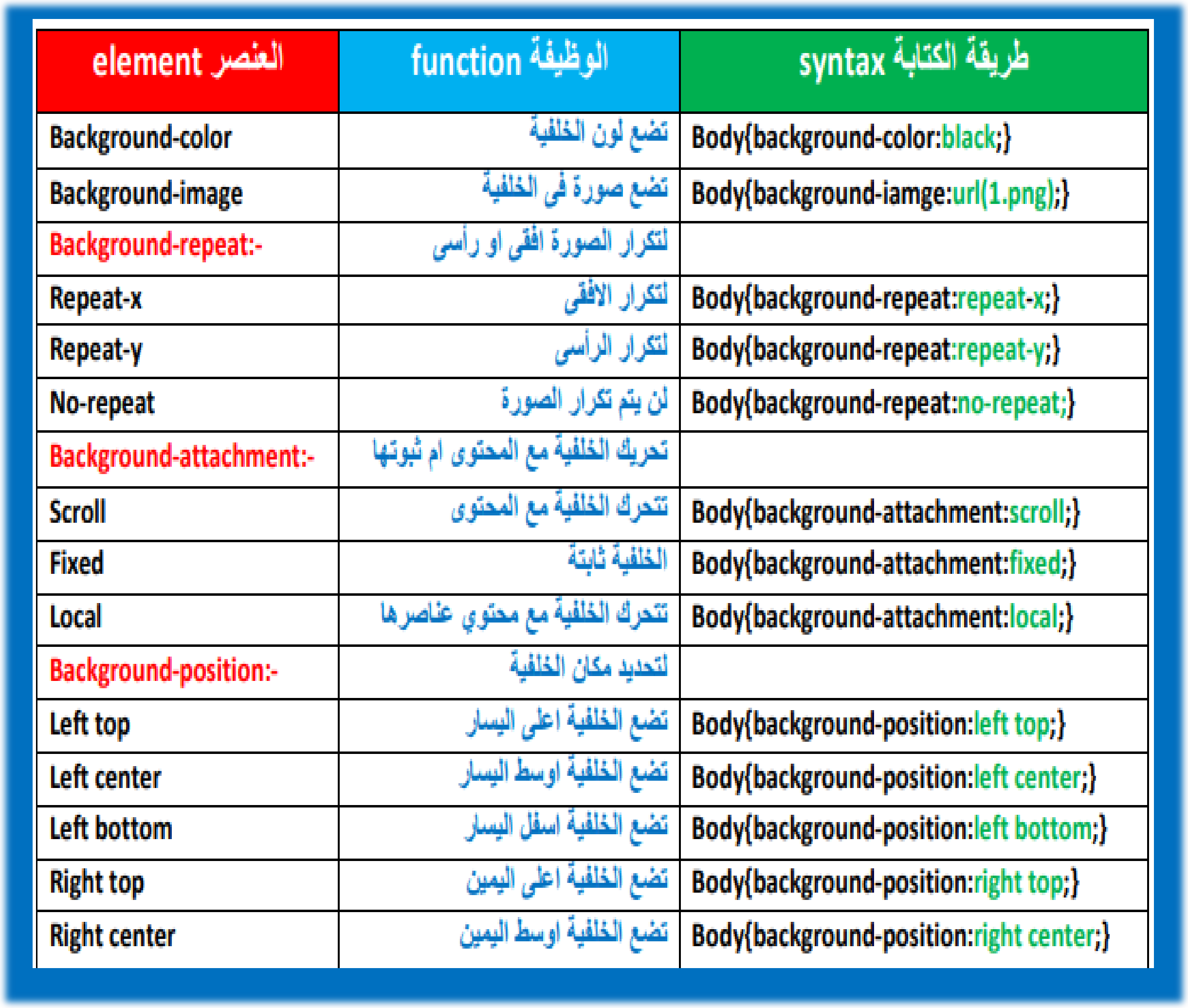
An external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is how the "styles.css" looks:

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

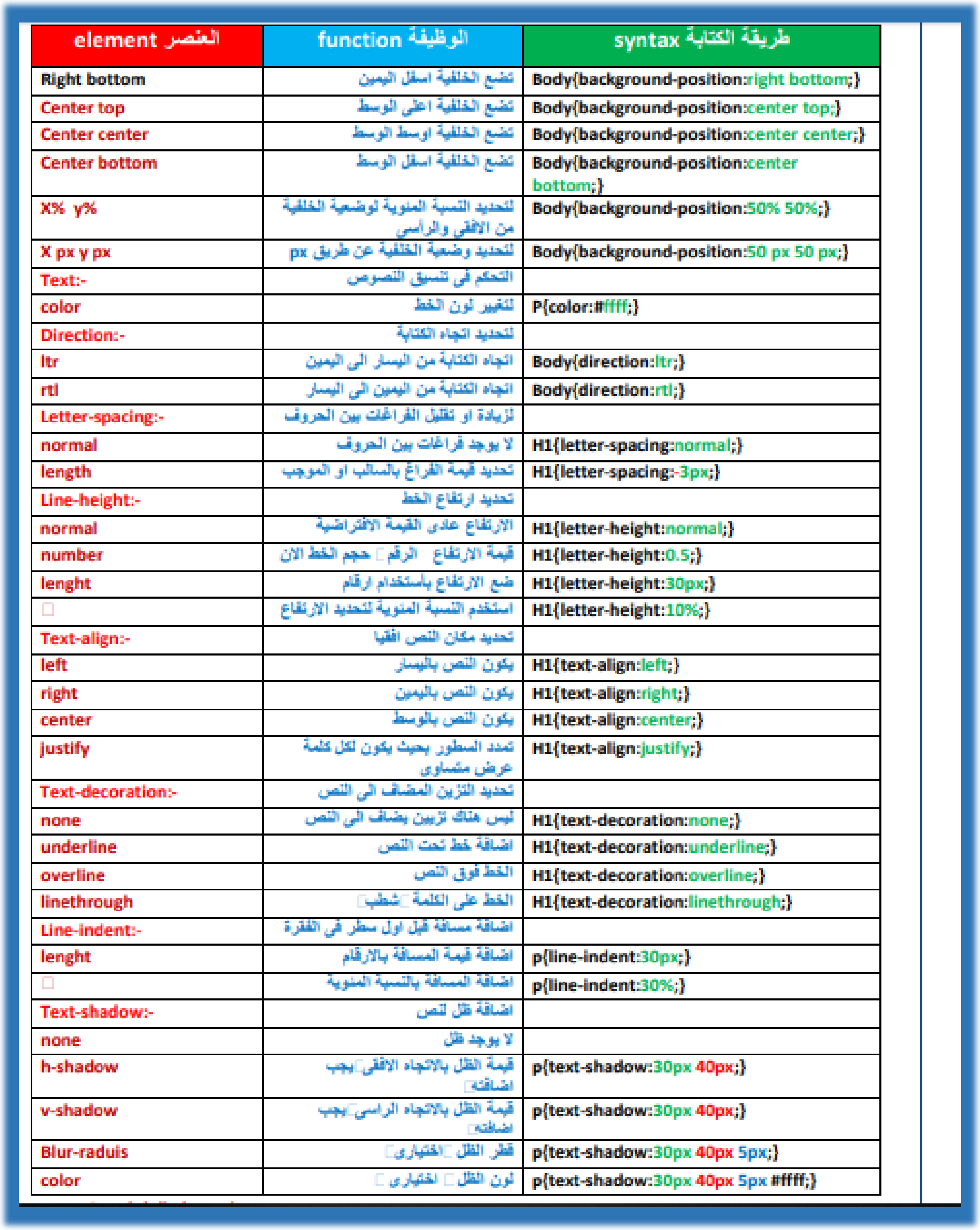
The elements and functions in CSS

Table 1:



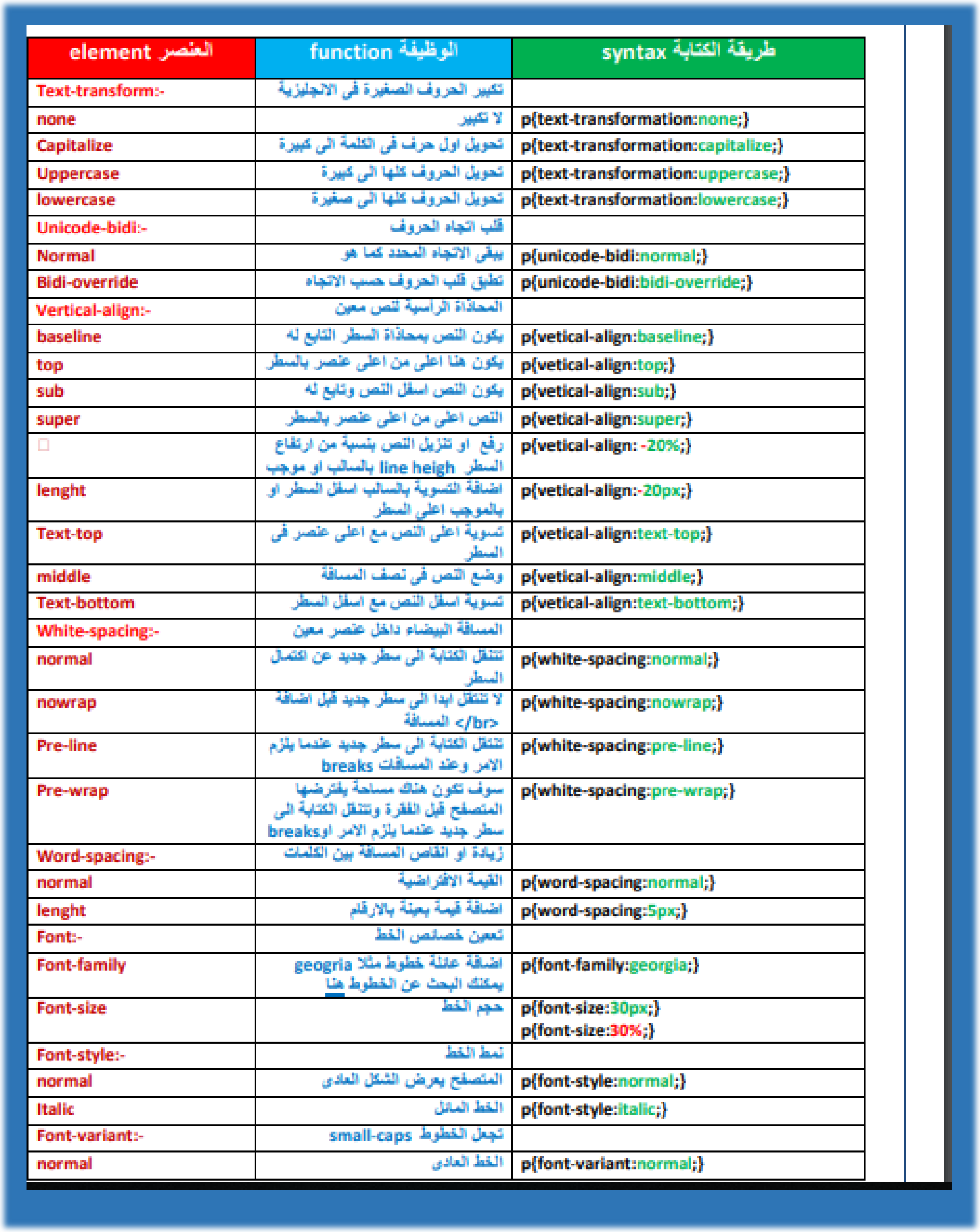
element العنصر	function الوظيفة	syntax طريقة الكتابة
Background-color	تضع لون الخلفية	Body{background-color:black;}
Background-image	تضع صورة في الخلفية	Body{background-iamge:url(1.png);}
Background-repeat:-	لتكرار الصورة افقي او رأسي	
Repeat-x	لتكرار الافقي	Body{background-repeat:repeat-x;}
Repeat-y	لتكرار الرأسي	Body{background-repeat:repeat-y;}
No-repeat	لن يتم تكرار الصورة	Body{background-repeat:no-repeat;}
Background-attachment:-	تحريك الخلفية مع المحتوى ام ثبوتها	
Scroll	تتحرك الخلفية مع المحتوى	Body{background-attachment:scroll;}
Fixed	الخلفية ثابتة	Body{background-attachment:fixed;}
Local	تتحرك الخلفية مع محتوى عناصرها	Body{background-attachment:local;}
Background-position:-	لتحديد مكان الخلفية	
Left top	تضع الخلفية اعلى اليسار	Body{background-position:left top;}
Left center	تضع الخلفية اوسط اليسار	Body{background-position:left center;}
Left bottom	تضع الخلفية اسفل اليسار	Body{background-position:left bottom;}
Right top	تضع الخلفية اعلى اليمين	Body{background-position:right top;}
Right center	تضع الخلفية اوسط اليمين	Body{background-position:right center;}

Table 2:



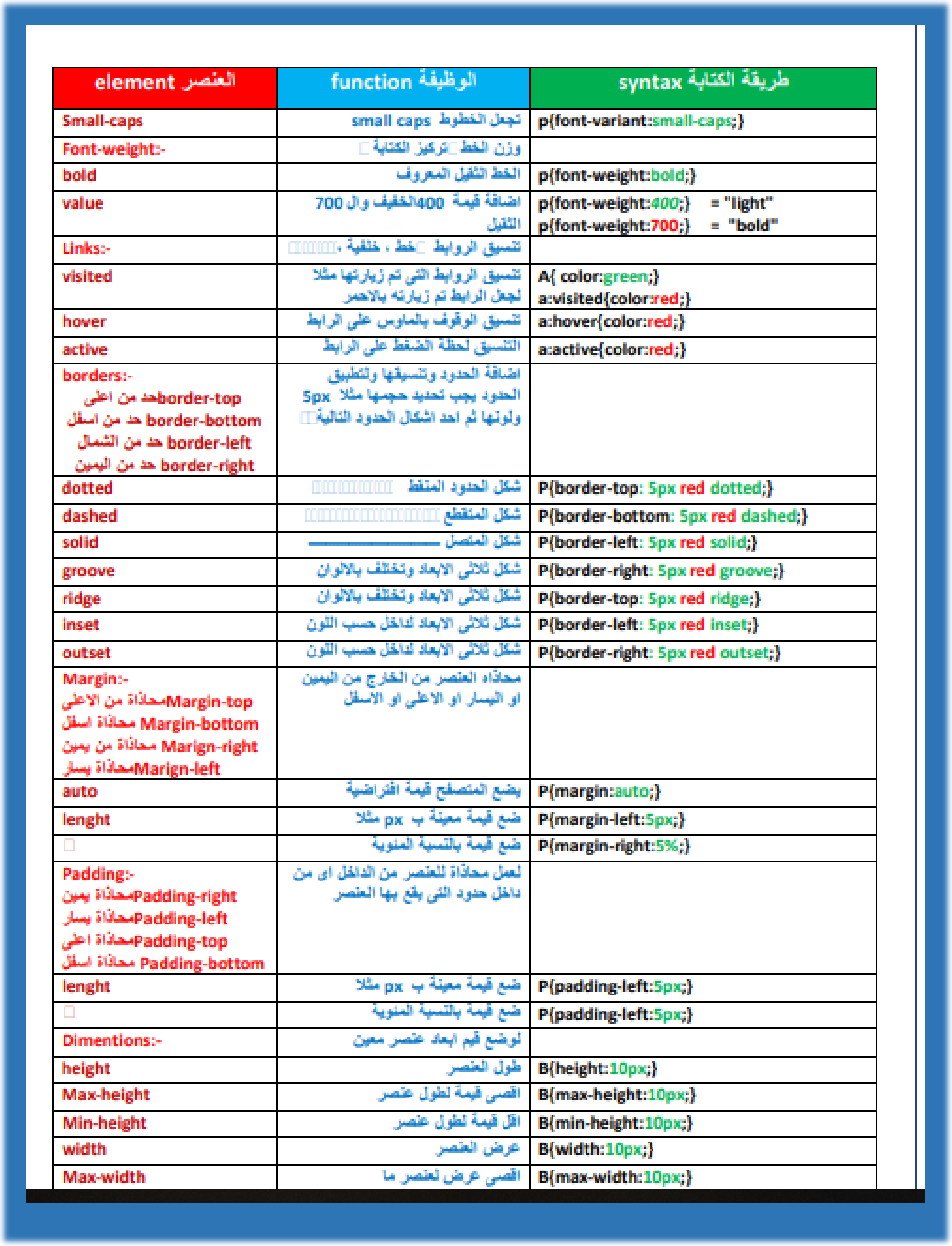
element العنصر	function الوظيفة	syntax طريقة الكتابة
Right bottom	تضع الخلفية اسفل اليمين	Body{background-position:right bottom;}
Center top	تضع الخلفية اعلى الوسط	Body{background-position:center top;}
Center center	تضع الخلفية اوسط الوسط	Body{background-position:center center;}
Center bottom	تضع الخلفية اسفل الوسط	Body{background-position:center bottom;}
X% y%	لتحديد النسبة المئوية لوضعية الخلفية من الافقي والراسي	Body{background-position:50% 50%;}
X px y px	لتحديد وضعية الخلفية عن طريق px	Body{background-position:50 px 50 px;}
Text:-	التحكم في تنسيق النصوص	
color	لتغيير لون الخط	P{color:#ffff;}
Direction:-	لتحديد اتجاه الكتابة	
ltr	اتجاه الكتابة من اليسار الى اليمين	Body{direction:ltr;}
rtl	اتجاه الكتابة من اليمين الى اليسار	Body{direction:rtl;}
Letter-spacing:-	لزيادة او تقليل الفراغات بين الحروف	
normal	لا يوجد فراغات بين الحروف	H1{letter-spacing:normal;}
length	تحديد قيمة الفراغ بالسالب او الموجب	H1{letter-spacing:-3px;}
Line-height:-	تحديد ارتفاع الخط	
normal	الارتفاع عادى القيمة الافتراضية	H1{letter-height:normal;}
number	قيمة الارتفاع الرقم □ حجم الخط الان	H1{letter-height:0.5;}
lenght	ضع الارتفاع بأستخدام ارقام	H1{letter-height:30px;}
□	استخدم النسبة المئوية لتحديد الارتفاع	H1{letter-height:10%;}
Text-align:-	تحديد مكان النص افقيا	
left	يكون النص باليسار	H1{text-align:left;}
right	يكون النص باليمين	H1{text-align:right;}
center	يكون النص بالوسط	H1{text-align:center;}
justify	تمدد السطور بحيث يكون لكل كلمة عرض متساوى	H1{text-align:justify;}
Text-decoration:-	تحديد التزين المضاف الى النص	
none	ليس هناك تزيين يضاف الى النص	H1{text-decoration:none;}
underline	اضافة خط تحت النص	H1{text-decoration:underline;}
overline	الخط فوق النص	H1{text-decoration:overline;}
linethrough	الخط على الكلمة□شطب□	H1{text-decoration:linethrough;}
Line-indent:-	اضافة مسافة قبل اول سطر فى الفقرة	
lenght	اضافة قيمة المسافة بالارقام	p{line-indent:30px;}
□	اضافة المسافة بالنسبة المئوية	p{line-indent:30%;}
Text-shadow:-	اضافة ظل لنص	
none	لا يوجد ظل	
h-shadow	قيمة الظل بالاتجاه الافقى□يجب اضافته□	p{text-shadow:30px 40px;}
v-shadow	قيمة الظل بالاتجاه الراسى□يجب اضافته□	p{text-shadow:30px 40px;}
Blur-raduis	قطر الظل □اختيارى□	p{text-shadow:30px 40px 5px;}
color	لون الظل □ اختيارى □	p{text-shadow:30px 40px 5px #ffff;}

Table 3:



element العنصر	function الوظيفة	syntax طريقة الكتابة
Text-transform:-	تكبير الحروف الصغيرة في الانجليزية	
none	لا تكبير	p{text-transformation:none;}
Capitalize	تحويل اول حرف في الكلمة الى كبيرة	p{text-transformation:capitalize;}
Uppercase	تحويل الحروف كلها الى كبيرة	p{text-transformation:uppercase;}
lowercase	تحويل الحروف كلها الى صغيرة	p{text-transformation:lowercase;}
Unicode-bidi:-	قلب اتجاه الحروف	
Normal	يبقى الاتجاه المحدد كما هو	p{unicode-bidi:normal;}
Bidi-override	تطبق قلب الحروف حسب الاتجاه	p{unicode-bidi:bidi-override;}
Vertical-align:-	المحاذاة الراسية لنص معين	
baseline	يكون النص بمحاذاة السطر التابع له	p{vetical-align:baseline;}
top	يكون هنا اعلى من اعلى عنصر بالسطر	p{vetical-align:top;}
sub	يكون النص اسفل النص وتابع له	p{vetical-align:sub;}
super	النص اعلى من اعلى عنصر بالسطر	p{vetical-align:super;}
□	رفع او تنزيل النص بنسبة من ارتفاع السطر line heigh بالسالب او موجب	p{vetical-align: -20%;}
lenght	اضافة التسوية بالسالب اسفل السطر او بالموجب اعلى السطر	p{vetical-align:-20px;}
Text-top	تسوية اعلى النص مع اعلى عنصر فى السطر	p{vetical-align:text-top;}
middle	وضع النص في نصف المسافة	p{vetical-align:middle;}
Text-bottom	تسوية اسفل النص مع اسفل السطر	p{vetical-align:text-bottom;}
White-spacing:-	المسافة البيضاء داخل عنصر معين	
normal	تنتقل الكتابة الى سطر جديد عن اكتمال السطر	p{white-spacing:normal;}
nowrap	لا تنتقل ابدا الى سطر جديد قبل اضافة </br> المسافة	p{white-spacing:nowrap;}
Pre-line	تنتقل الكتابة الى سطر جديد عندما يلزم الامر وعند المسافات breaks	p{white-spacing:pre-line;}
Pre-wrap	سوف تكون هناك مساحة يفترضها المتصفح قبل الفقرة وتنتقل الكتابة الى سطر جديد عندما يلزم الامر اوbreaks	p{white-spacing:pre-wrap;}
Word-spacing:-	زيادة او انقاص المسافة بين الكلمات	
normal	القيمة الافتراضية	p{word-spacing:normal;}
lenght	اضافة قيمة بعينة بالارقام	p{word-spacing:5px;}
Font:-	تعين خصائص الخط	
Font-family	اضافة عائلة خطوط مثلا geogria يمكنك البحث عن الخطوط هنا	p{font-family:georgia;}
Font-size	حجم الخط	p{font-size:30px;} p{font-size:30%;}
Font-style:-	نمط الخط	
normal	المتصفح يعرض الشكل العادى	p{font-style:normal;}
Italic	الخط المائل	p{font-style:italic;}
Font-variant:-	تجعل الخطوط small-caps	
normal	الخط العادى	p{font-variant:normal;}

Table 4:

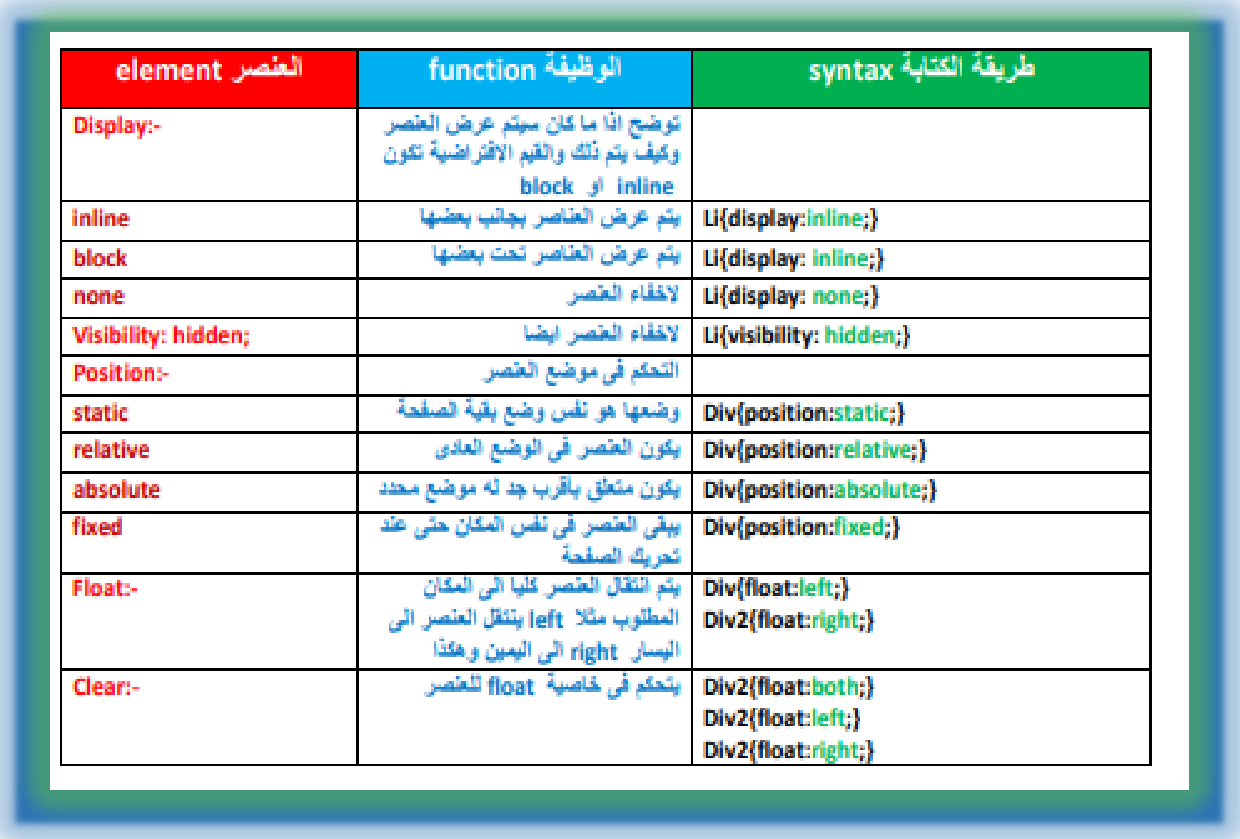


element العنصر	function الوظيفة	syntax طريقة الكتابة
Small-caps	تجعل الخطوط small caps	p{font-variant:small-caps;}
Font-weight:-	وزن الخط تركيز الكتابة	
bold	الخط الثقيل المعروف	p{font-weight:bold;}
value	اضافة قيمة 400الخفيف وال 700 الثقيل	p{font-weight:400;} = "light" p{font-weight:700;} = "bold"
Links:-	تنسيق الروابط خط ، خلفية ،	

visited	تنسيق الروابط التي تم زيارتها مثلا لجعل الرابط تم زيارته بالاحمر	A{ color:green;}
		a:visited{color:red;}

hover	تنسيق الوقوف بالماوس على الرابط	a:hover{color:red;}
active	التنسيق لحظة الضغط على الرابط	a:active{color:red;}
borders:- حد من اعلىborder-top حد من اسفل border-bottom حد من الشمال border-left حد من اليمين border-right	اضافة الحدود وتنسيقها ولتطبيق الحدود يجب تحديد حجمها مثلا 5px ولونها ثم احد اشكال الحدود التالية	
dotted	شكل الحدود المنقط	P{border-top: 5px red dotted;}
dashed	شكل المتقطع	P{border-bottom: 5px red dashed;}
solid	شكل المتصل ————————	P{border-left: 5px red solid;}
groove	شكل ثلاثي الابعاد وتختلف بالالوان	P{border-right: 5px red groove;}
ridge	شكل ثلاثي الابعاد وتختلف بالالوان	P{border-top: 5px red ridge;}
inset	شكل ثلاثي الابعاد لداخل حسب اللون	P{border-left: 5px red inset;}
outset	شكل ثلاثي الابعاد لداخل حسب اللون	P{border-right: 5px red outset;}
Margin:- محاذاة من الاعلىMargin-top محاذاة اسفل Margin-bottom محاذاة من يمين Marign-right محاذاة يسارMarign-left	محاذاه العنصر من الخارج من اليمين او اليسار او الاعلى او الاسفل	
auto	يضع المتصفح قيمة افتراضية	P{margin:auto;}
lenght	ضع قيمة معينة ب px مثلا	P{margin-left:5px;}
	ضع قيمة بالنسبة المئوية	P{margin-right:5%;}
Padding:- محاذاة يمينPadding-right محاذاة يسارPadding-left محاذاة اعلىPadding-top محاذاة اسفل Padding-bottom	لعمل محاذاة للعنصر من الداخل اى من داخل حدود التي يقع بها العنصر	
lenght	ضع قيمة معينة ب px مثلا	P{padding-left:5px;}
	ضع قيمة بالنسبة المئوية	P{padding-left:5px;}
Dimentions:-	لوضع قيم ابعاد عنصر معين	
height	طول العنصر	B{height:10px;}
Max-height	اقصى قيمة لطول عنصر	B{max-height:10px;}
Min-height	اقل قيمة لطول عنصر	B{min-height:10px;}
width	عرض العنصر	B{width:10px;}
Max-width	اقصى عرض لعنصر ما	B{max-width:10px;}

Table 5:



element العنصر	function الوظيفة	syntax طريقة الكتابة
Display:-	توضح اذا ما كان سيتم عرض العنصر وكيف يتم ذلك والقيم الافتراضية تكون inline او block	
inline	يتم عرض العناصر بجانب بعضها	Li{display:inline;}
block	يتم عرض العناصر تحت بعضها	Li{display: inline;}
none	لاخفاء العنصر	Li{display: none;}
Visibility: hidden;	لاخفاء العنصر ايضا	Li{visibility: hidden;}
Position:-	التحكم في موضع العنصر	
static	وضعها هو نفس وضع بقية الصفحة	Div{position:static;}
relative	يكون العنصر في الوضع العادى	Div{position:relative;}
absolute	يكون متعلق بأقرب جد له موضع محدد	Div{position:absolute;}
fixed	يبقى العنصر في نفس المكان حتى عند تحريك الصفحة	Div{position:fixed;}

Float:-	يتم انتقال العنصر كليا الى المكان المطلوب مثلا left ينتقل العنصر الى اليسار right الى اليمين وهكذا	Div{float:left;}
		Div2{float:right;}

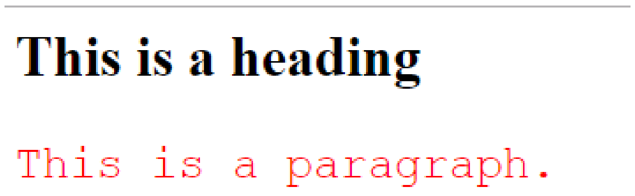
Clear:-	يتحكم في خاصية float للعنصر	Div2{float:both;}
		Div2{float:left;}
		Div2{float:right;}

CSS: Examples:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>                              css
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
```

```
p  {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>

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

</body>
</html>
```



This is a heading

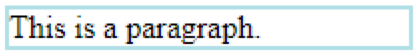
This is a paragraph.

CSS Border

The CSS border property defines a border around an HTML element:

Example

```
<style>

p {
  border: 1px solid powderblue;

width:200px;
}
</style>
```



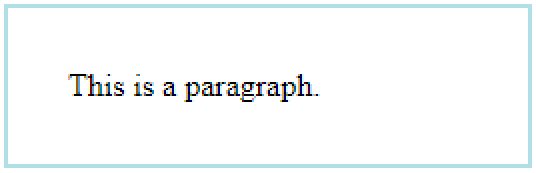
This is a paragraph.

CSS Padding

The CSS padding property defines a padding (space) between the text and the border:

Example

```
p {
  border: 1px solid powderblue;

  width=300px;
  padding: 30px;
}

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



CSS Margin

The CSS margin property defines a margin (space) outside the border:

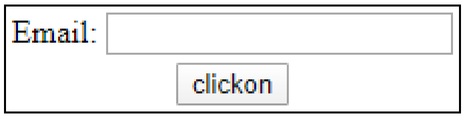
Example

```
p {
  border: 1px solid powderblue;
  margin: 50px;
}
```

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

Example: table

```
<style>

Table {

border:1px solid black;

}

</style>

<table>

<tr>

<td>Email:</td>

<td><input type="text" value"enter your email/></td></br>

</tr>

<tr>

<th colspan="2" ><input type="button" value="clickon" /></th></br>

</tr>

</table>
```



border only for table

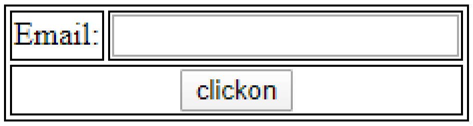
Ex:

<style>

Table ,td,th{

border:1px solid black;

}



border for table and cells

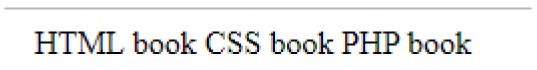
Example:list

<style>

li{

display:inline;

```
}
</style>
<body>
<ul>
<li>HTML book</li>
<li>CSS book</li>
<li>PHP book</li>
</ul>
</body>
```

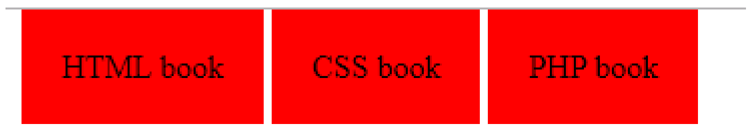


```
HTML book CSS book PHP book
```

inline

Ex:

```
<style>
li{
display:inline;
background-color:red;
padding:20px;
}
</style>
```



Ex:<style>

ul{

padding-top:20px;

}

li{

display:inline;

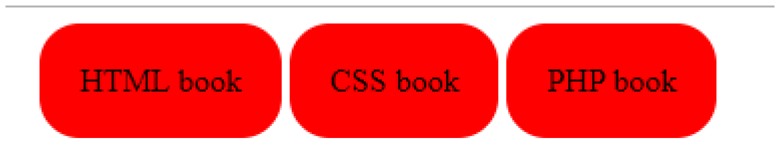
background-color:red;

padding:20px;

border-radius:20px;

}

</style>



The(id) Attribute

To define a specific style for one special element, add an id attribute to the element:

<body>

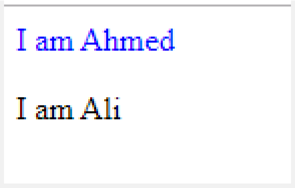
```
<p id="pp">I am Ahmed</p>
<p > I am Ali</p>
</body>
```

then define a style for the element with the specific id:

Example

```
<style>
#pp {
  color: blue;
}
</style>
```

فقط العنصر الذي وضعنه له idيصبح لونه ازرق



```
I am Ahmed

I am Ali
```

Note: The id of an element should be unique within a page, so the id selector is used to select one unique element!

The(class)Attribute

To define a style for special types of elements, add a class attribute to the element:

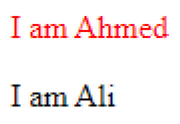
```
<p class="error">I am Ahmed</p>
```

```
<p>I am Ali</p>
```

then define a style for the elements with the specific class:

ex:

```
.error{
  color: red;
}
```

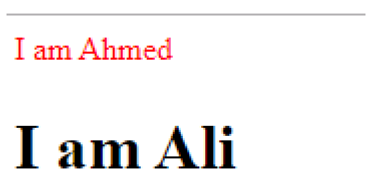

I am Ahmed

I am Ali

```
<p class=”error”>I am Ahmed</p>
<h1 class=”error”>I am Ali</h1>
```

Example

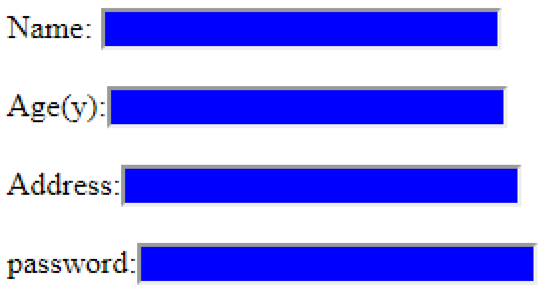
```
p.error {
  color: red;
}
```


I am Ahmed

I am Ali

Example: form

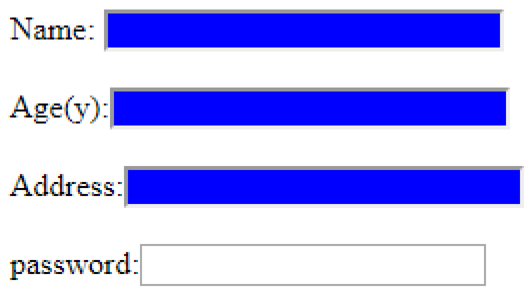
```
<style>
input {
  color:white;
background-color:blue;
width:200px;
}
</style>
<body>
Name:   <input type="text"/></br></br>
Age(y):<input type="text"/></br></br>
Address:<input type="text"/></br></br>
password:<input type="password"/></br>


</body>
```



```
Name:
Age(y):
Address:
password:
```

Ex:<style>

```
input[type="text"]{
  color:white;
background-color:blue;
width:200px;
}
</style>
```


Name:

Age(y):

Address:

password:

JavaScript

JavaScript is the programming language of HTML and the Web.

JavaScript is easy to learn.

JavaScript Can Change HTML Content

One of many JavaScript HTML methods is getElementById().

The <script> Tag

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

Example::مثال

```
<script>
</script>
<body>
<p >A Paragraph</p>
</body>
```

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 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>

In this example, a JavaScript is placed in the <head> section of an HTML page.

Example

```
<!DOCTYPE html>
<html>

<script>
              الايعازات
</script>

<body>
<h1>A Web Page</h1>
</body>
</html>
```

JavaScript in <body>

In this example, a JavaScript is placed in the <body> section of an HTML page.

Example

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

<h1>A Web Page</h1>
<script>
الايعازات

</script>
</body>
</html>
```

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

External JavaScript مكانها بملف اخر

Scripts can also be placed in external files.

JavaScript files have the file extension .js.(**myScript.js**).

External scripts cannot contain <script> tags.

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

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

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 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>
```

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Using innerHTML

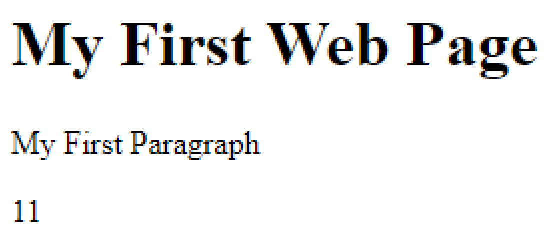
To access an HTML element, JavaScript can use the document.getElementById(id) method. The id attribute defines the HTML element.

document.getElementById(id) . صفه =

The innerHTML property defines the HTML content:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="pp1"></p>
<script>
document.getElementById("pp1").innerHTML = 5 + 6;

</script>
</body>
</html>
```



```
My First Web Page

My First Paragraph

11
```

Changing the innerHTML property of an HTML element is a common way to display data in HTML.

الصفه الثانيه : style   (style.display, style.backgrond,…. مثلا)

document.getElementById("pp").style.display='none';

document.getElementById("pp").style.display='block';

الصفه الثالثه :value

document.getElementById("pp").value

Example:

Number1 :<input type="text " id=”pp”/>

number1 :

++

الصفه الرابعه: checked

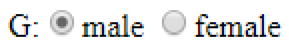
اذا اردنا القيمه التي تم اختيارها من option or check

ب option يكون خيار واحد فقط لذلك نستخدم if else

Example

G:<input type="radio" id="m"/>male

<input type="radio" id="f"/>female


G: ● male ○ female

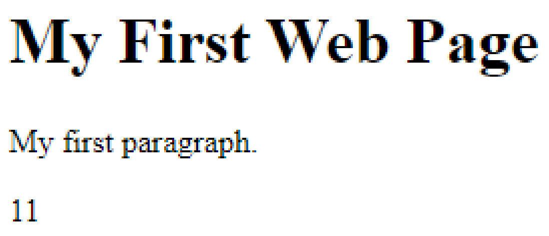
```
<script>
If (document.getElementById("m").checked=true)
    alert (“male”);
else
alert(“female”);
</script>
```

++
+Using document.write()

For testing purposes, it is convenient to use document.write():

Example

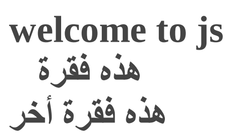
```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
document.write(5 + 6);
</script>
</body>
</html>
```



```
My First Web Page

My first paragraph.

11
```

Example:

```
<script>
  document.write("<h1>welcome to js </h1>");
  document.write("<p> هذه فقرة </p>");
  document.write("<p> هذه فقرة أخرى </p>");
</script>
```



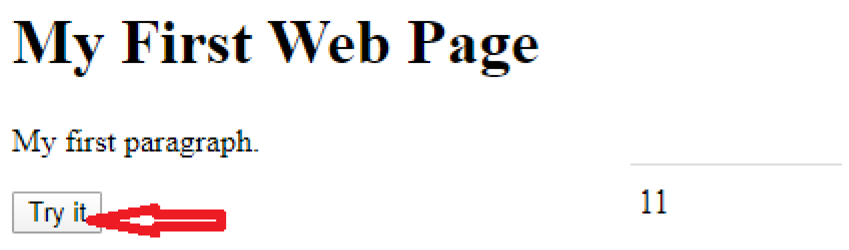
```
welcome to js
هذه فقرة
هذه فقرة أخر
```

Using document.write() after an HTML document is loaded, will delete all existing HTML:

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>
```

```
<button type="button" onclick="document.write(5 + 6)">Try it</button>

</body>
</html>
```



My First Web Page

My first paragraph.

Try it

11

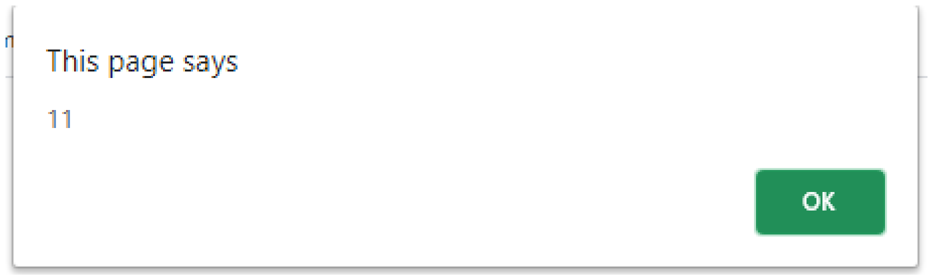
+++

Using window.alert()

You can use an alert box to display data:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);


</script>
</body>
</html>
```



This page says

11

++

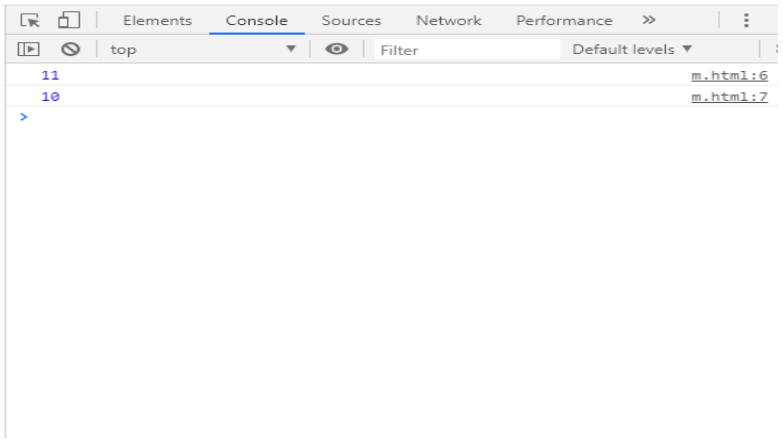
Using console.log()

For debugging purposes, you can use the `console.log()` method to display data.

Example

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

<script>
console.log(5 + 6);

console.log(5 *2);
</script>

</body>
</html>
```

تعتبر هذه الطريقه من اهم الطرق وافضلها .النتائج تظهر باستخدام debug الموجود باي متصفح(ضغط F12) ومن ثم نختار console



```
11                                                    m.html:6
10                                                    m.html:7
>
```


JavaScript Statements

Example

```
var x, y, z;    // Statement 1
x = 5;          // Statement 2
```

```
y = 6;          // Statement 3
z = x + y;      // Statement 4
```

JavaScript Programs

A computer program is a list of "instructions" to be "executed" by a computer.

In a programming language, these programming instructions are called statements.

A JavaScript program is a list of programming statements.

In HTML, JavaScript programs are executed by the web browser.

JavaScript Statements

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="pp":

Example

```
document.getElementById("pp").innerHTML = "Hello Dolly.";
```

Most JavaScript programs contain many JavaScript statements.

The statements are executed, one by one, in the same order as they are written.

JavaScript programs (and JavaScript statements) are often called JavaScript code.

Semicolons ;

Semicolons separate JavaScript statements.

Add a semicolon at the end of each executable statement:

```
var a, b, c;     // Declare 3 variables
a = 5;           // Assign the value 5 to a
b = 6;           // Assign the value 6 to b
c = a + b;       // Assign the sum of a and b to c
```

When separated by semicolons, multiple statements on one line are allowed:

```
a = 5; b = 6; c = a + b;
```

On the web, you might see examples without semicolons.
Ending statements with semicolon is not required, but highly recommended.

JavaScript White Space

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

The following lines are equivalent:

```
var person = "Hege";
var person="Hege";
```

A good practice is to put spaces around operators (= + - * /):

```
var x = y + z;
```

JavaScript Line Length and Line Breaks

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it is after an operator:

Example

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

JavaScript Code Blocks

JavaScript statements can be grouped together in code blocks, inside curly brackets
{…….

}

The purpose of code blocks is to define statements to be executed together.

One place you will find statements grouped together in blocks, is in JavaScript functions:

Example

```
function myFunction() {
  document.getElementById("demo1").innerHTML = "Hello Dolly!";
  document.getElementById("demo2").innerHTML = "How are you?";
}
```

JavaScript Keywords

JavaScript statements often start with a keyword to identify the JavaScript action to be performed.

Visit our Reserved Words reference to view a full list of JavaScript keywords.

Here is a list of some of the keywords you will learn about in this tutorial:

Keyword	Description
break	Terminates a switch or a loop

continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
For	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
Var	Declares a variable

JavaScript keywords are reserved words. Reserved words cannot be used as names for variables.

++

JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

```
var x, y, z;       // How to declare variables
x = 5; y = 6;      // How to assign values
z = x + y;         // How to compute values
```

JavaScript Values

The JavaScript syntax defines two types of values: Fixed values and variable values.

Fixed values are called literals. Variable values are called variables.

JavaScript Literals

The most important rules for writing fixed values are:

Numbers are written with or without decimals:

10.50

1001

Strings are text, written within double or single quotes:

"John Doe"

'John Doe'

JavaScript Variables

In a programming language, variables are used to store data values.

JavaScript uses the `var` keyword to declare variables.

An equal sign is used to assign values to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

var x;

x = 6;

JavaScript Operators

JavaScript uses arithmetic operators (+ - * /) to compute values:

(5 + 6) * 10

JavaScript uses an assignment operator (=) to assign values to variables:

var x, y;
x = 5;
y = 6;

JavaScript Expressions

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 * 10 evaluates to 50:

Expressions can also contain variable values:

x * 10

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

John1

John2

John3

ب html يكون :

```
John1</br>

John2</br>

John3</br>
```

javascriptب

```
<div id="bb"> </div>                    html عنصر

<script>                                هنا نكتب

X=" ";

For (i=1;i<=3;i++)

X+=" John"+i+"</br>"

document.getElementById("bb").innerHTML=x;

</script>
```

++

مثال اخر

```
<div id="bb"></div>

<script>

X=" ";

x="<h1>welcome</h1></br>":

x+="<h2>hellow</h2>";

document.getElementById("bb").innerHTML=x;
```

<script/>

JavaScript Comments

Not all JavaScript statements are "executed".

Code after double slashes // or between /* and */ is treated as a comment.

Comments are ignored, and will not be executed:

var x = 5;   // I will be executed

// var x = 6;   I will NOT be executed

JavaScript Identifiers

Identifiers are names.

In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).

The rules for legal names are much the same in most programming languages.

In JavaScript, the first character must be a letter, or an underscore (_), or a dollar sign ($).

Subsequent characters may be letters, digits, underscores, or dollar signs.

Numbers are not allowed as the first character.
This way JavaScript can easily distinguish identifiers from numbers.

JavaScript is Case Sensitive

All JavaScript identifiers are case sensitive.

The variables lastName and lastname, are two different variables:

```
var lastname, lastName;
lastName = "Doe";
lastname = "Peterson";
```

JavaScript does not interpret VAR or Var as the keyword var.

JavaScript Data Types

JavaScript variables can hold numbers like 100 and text values like "John Doe".

In programming, text values are called text strings.

JavaScript can handle many types of data, but for now, just think of numbers and strings.

Strings are written inside double or single quotes. Numbers are written without quotes.

If you put a number in quotes, it will be treated as a text string.

Example

```
var pi = 3.14;
var person = "John Doe";
var answer = 'Yes I am!';
```

Declaring (Creating) JavaScript Variables

Creating a variable in JavaScript is called "declaring" a variable.

You declare a JavaScript variable with the `var` keyword:

```
var carName;
```

After the declaration, the variable has no value (technically it has the value of `undefined`).

To assign a value to the variable, use the equal sign:

```
carName = "Volvo";
```

You can also assign a value to the variable when you declare it:

```
var carName = "Volvo";
```

In the example below, we create a variable called `carName` and assign the value "Volvo" to it.

Then we "output" the value inside an HTML paragraph with id="pp":

Example

```
<p id="pp"></p>
<script>
var carName = "Volvo";
document.getElementById("pp").innerHTML = carName;
</script>
```

It's a good programming practice to declare all variables at the beginning of a script.

One Statement, Many Variables

You can declare many variables in one statement.

Start the statement with `var` and separate the variables by comma:

```
var person = "John Doe", carName = "Volvo", price = 200;
```

A declaration can span multiple lines:

```
var person = "John Doe",
carName = "Volvo",
price = 200;
```

A variable declared without a value will have the value `undefined`.

The variable carName will have the value `undefined` after the execution of this statement:

Example

```
var carName;
```

JavaScript Arithmetic

As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +:

Example

```
var x = 5 + 2 + 3;
```

You can also add strings, but strings will be concatenated:

Example

```
var x = "John" + " " + "Doe";
```

Also try this:

Example

```
var x = "5" + 2 + 3;
```

If you put a number in quotes, the rest of the numbers will be treated as strings, and concatenated.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description
+	Addition

-	**Subtraction**
*****	**Multiplication**
******	**Exponentiation (ES2016)**
/	**Division**
%	**Modulus (Division Remainder)**
++	**Increment**
--	**Decrement**

Arithmetic operators are fully described in the JS Arithmetic chapter.

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As

=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x – y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

Example

```
var txt1 = "John";
var txt2 = "Doe";
var txt3 = txt1 + " " + txt2;
```

The result of txt3 will be:

John Doe

The += assignment operator can also be used to add (concatenate) strings:

Example

```
var txt1 = "What a very ";
txt1 += "nice day";
```

The result of txt1 will be:

What a very nice day

When used on strings, the + operator is called the concatenation operator.

Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string:

Example

```
var x = 5 + 5;
var y = "5" + 5;
var z = "Hello" + 5;
```

The result of *x*, *y*, and *z* will be:

```
10
55
Hello5
```

If you add a number and a string, the result will be a string!

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Comparison operators are fully described in the JS Comparisons **chapter.**

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

Logical operators are fully described in the JS Comparisons chapter.

JavaScript Type Operators

Operator	Description
Typeof	Returns the type of a variable
Instanceof	Returns true if an object is an instance of an object type

JavaScript Booleans

Booleans can only have two values: true or false.

Example

```
var x = 5;
var y = 5;
var z = 6;
(x == y)      // Returns true
(x == z)      // Returns false
```

Booleans are often used in conditional testing.

You will learn more about conditional testing later in this tutorial.

JavaScript Objects

JavaScript objects are written with curly braces {}.

Object properties are written as name:value pairs, separated by commas.

Example

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.

You will learn more about objects later in this tutorial.

The typeof Operator

You can use the JavaScript typeof operator to find the type of a JavaScript variable.

The typeof operator returns the type of a variable or an expression:

Example

```
typeof ""            // Returns "string"
typeof "John"        // Returns "string"
typeof "John Doe"    // Returns "string"
```

Example

```
typeof 0             // Returns "number"
typeof 314           // Returns "number"
typeof 3.14          // Returns "number"
typeof (3)           // Returns "number"
typeof (3 + 4)       // Returns "number"
```

Empty Values

An empty value has nothing to do with `undefined`.

An empty string has both a legal value and a type.

Example

```
var car = "";    // The value is "", the typeof is "string"
```

Null and undefined

In JavaScript `null` is "nothing". It is supposed to be something that doesn't exist.

Unfortunately, in JavaScript, the data type of `null` is an object.

You can consider it a bug in JavaScript that `typeof null` is an object. It should be `null`.

You can empty an object by setting it to `null`:

Example

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
person = null;    // Now value is null, but type is still an object
```

You can also empty an object by setting it to `undefined`:

Example

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
person = undefined;   // Now both value and type is undefined
```

Difference Between Undefined and Null

undefined and null are equal in value but different in type:

```
typeof undefined          // undefined
typeof null               // object

null === undefined        // false           النوع
null == undefined         // true            القيمه
```

Primitive Data

A primitive data value is a single simple data value with no additional properties and methods.

The typeof operator can return one of these primitive types:

- string
- number
- boolean
- undefined

Example

```
typeof "John"          // Returns "string"
typeof 3.14            // Returns "number"
typeof true            // Returns "boolean"
typeof false           // Returns "boolean"
typeof x               // Returns "undefined" (if x has no value)
```

Complex Data

The typeof operator can return one of two complex types:

- function
- object

The typeof operator returns "object" for objects, arrays, and null.

The typeof operator does not return "object" for functions.

Example

```
typeof {name:'John', age:34} // Returns "object"
typeof [1,2,3,4]            // Returns "object" (not "array", see note below)
typeof null                 // Returns "object"
typeof function myFunc(){}   // Returns "function"
```

++

JavaScript Functions

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:
(*parameter1, parameter2, ...*)

The code to be executed, by the function, is placed inside curly brackets: {}

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Function parameters are listed inside the parentheses () in the function definition.

Function arguments are the values received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

A Function is much the same as a Procedure or a Subroutine, in other programming languages.

Function Invocationالاستدعاء

The code inside the function will execute when "something" invokes (calls) the function:

- **When an event occurs (when a user clicks a button)**
- **When it is invoked (called) from JavaScript code**
- **Automatically (self invoked)**

You will learn a lot more about function invocation later in this tutorial.

Function Return

When JavaScript reaches a `return` statement, the function will stop executing.

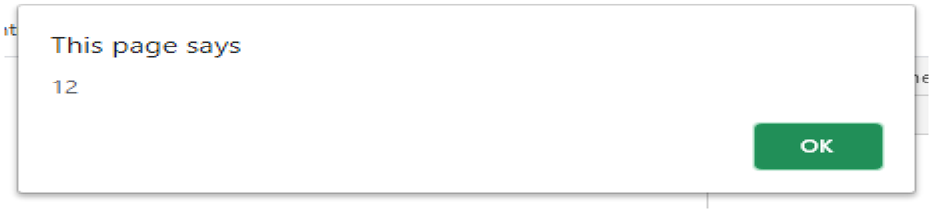
If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a return value. The return value is "returned" back to the "caller":

Example:Calculate the product of two numbers, and return the result:

```
function myFunction(a, b) {
  return a * b;             // Function returns the product of a and b
}



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

<script>

var x = myFunction(4, 3);   // Function is called, return value will end up in x
```

```
 alert(x);
</script>
```



This page says
12

Example:

```
<script>
function myFunction(p1, p2) {
  return p1 * p2;  }    // The function returns the product of p1 and p2
</script>
<body>
<script>
var x = myFunction(4, 3);
alert(x);
</script>     </body>
<script>
function myFunction(p1, p2) {
  return p1 * p2;   }// The function returns the product of p1 and p2
</script>
<body>
```

```
<input type="button"  value="result" onclick="alert(myFunction(4, 3))"/>

</body>
```

Why Functions?

You can reuse code: Define the code once, and use it many times.

You can use the same code many times with different arguments, to produce different results.

Example :Convert Fahrenheit to Celsius:

```
<div id="pp"></div>

<script>

function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
document.getElementById("pp").innerHTML = toCelsius(77);

</script>
```

Functions Used as Variable Values

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.

Example

Instead of using a variable to store the return value of a function:

```
var x = toCelsius(77);
var text = "The temperature is " + x + " Celsius";
```

You can use the function directly, as a variable value:

```
var text = "The temperature is " + toCelsius(77) + " Celsius";
```

You will learn a lot more about functions later in this tutorial.

Local Variables تستخدم داخل الداله

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

Local variables can only be accessed from within the function.

Example

```
// code here can NOT use carName

function myFunction() {
  var carName = "Volvo";
  // code here CAN use carName
}

// code here can NOT use carName
```

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

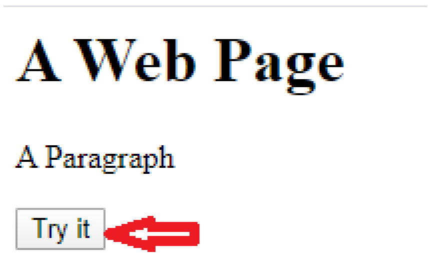
Local variables are created when a function starts, and deleted when the function is completed.

Example:

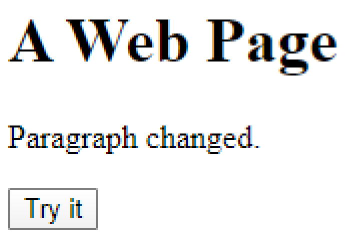
```
<!DOCTYPE html>
<html>

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




</head>
<body>
```

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

</body>
</html>
```

التنفيذ


A Web Page

A Paragraph


A Web Page

Paragraph changed.

++

JavaScript Arrays

JavaScript arrays are used to store multiple values in a single variable.

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

var *array_name* = [*item1*, *item2*, ...];   //one

var *array_name* = [[*item1*, *item2*, ...], [[*item1*, *item2*, ...], [[*item1*, *item2*, ...]…]; //two

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

Spaces and line breaks are not important. A declaration can span multiple lines:

Example

```
var cars = [
  "Saab",
  "Volvo",
  "BMW"
];
```

Using the JavaScript Keyword new

The following example also creates an Array, and assigns values to it:

Example

```
var cars = new Array("Saab", "Volvo", "BMW");//one
var c=new Array;
c[0]=new Array("Saab", "Volvo", "BMW");          //two
c[1]=new Array("Saab", "Volvo", "BMW");
```

The two examples above do exactly the same. There is no need to use new Array(). For simplicity, readability and execution speed, use the first one (the array literal method).

```
 <script>

var i;

var cars=new Array;

for(i=0;i<4;i++)

 cars[i]=prompt('enter the values of  array ');

alert (cars);

</script>
```

++

Access the Elements of an Array

You access an array element by referring to the index number.

This statement accesses the value of the first element in cars:

```
var name = cars[0];
```

Example

```
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("pp").innerHTML = cars[0];
```

Note: Array indexes start with 0.

[0] is the first element. [1] is the second element.

Changing an Array Element

This statement changes the value of the first element in `cars`:

```
cars[0] = "Opel";
```

Example

```
var cars = ["Saab", "Volvo", "BMW"];
cars[0] = "Opel";
document.getElementById("demo").innerHTML = cars[0];
```

Access the Full Array

With JavaScript, the full array can be accessed by referring to the array name:

Example

```
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;
```

Array Properties and Methods

The real strength of JavaScript arrays are the built-in array properties and methods:

Examples

```
var x = cars.length;   // The length property returns the number of elements
var y = cars.sort();   // The sort() method sorts arrays
```

The length Property

The `length` property of an array returns the length of an array (the number of array elements).

Example

var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.length;   // the length of fruits is 4

The length property is always one more than the highest array index.

Accessing the First Array Element

Example

fruits = ["Banana", "Orange", "Apple", "Mango"];
var first = fruits[0];

Accessing the Last Array Element

Example

fruits = ["Banana", "Orange", "Apple", "Mango"];
var last = fruits[fruits.length - 1];

++

javaScript if else and else if

Conditional statements are used to perform different actions based on different conditions.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **Use if to specify a block of code to be executed, if a specified condition is true**
- **Use else to specify a block of code to be executed, if the same condition is false**
- **Use else if to specify a new condition to test, if the first condition is false**

- **Use switch to specify many alternative blocks of code to be executed**

The if Statement

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

```
if (condition) {
  //  block of code to be executed if the condition is true
}
```

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error.

Example

Make a "Good day" greeting if the hour is less than 18:00:

```
if (hour < 18) {
  greeting = "Good day";
}
```

Example:

```
<script>

u=prompt('enter your name');

if ((u="Ali")

alert("welcome");

</script>
```

The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

```
if (condition) {
  //  block of code to be executed if the condition is true
} else {
  //  block of code to be executed if the condition is false
}
```

Example

If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

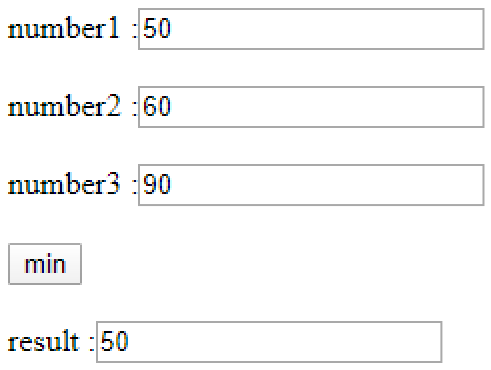
Syntax

```
if (condition1) {
  //  block of code to be executed if condition1 is true
} else if (condition2) {
  //  block of code to be executed if the condition1 is false and condition2 is true
} else {
  //  block of code to be executed if the condition1 is false and condition2 is false
}
```

Example

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```


number1 :50

number2 :60

number3 :90

result :50

++

JavaScript For Loop

Loops can execute a block of code a number of times.

JavaScript Loops

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

Instead of writing:

```
text += cars[0] + "<br>";
text += cars[1] + "<br>";
text += cars[2] + "<br>";
text += cars[3] + "<br>";
text += cars[4] + "<br>";
text += cars[5] + "<br>";
```

You can write:

```
var i;
for (i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
```

Different Kinds of Loops

JavaScript supports different kinds of loops:

- **for - loops through a block of code a number of times**
- **for/in - loops through the properties of an object**
- **for/of - loops through the values of an iterable object**
- **while - loops through a block of code while a specified condition is true**
- **do/while - also loops through a block of code while a specified condition is true**

The For Loop

The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

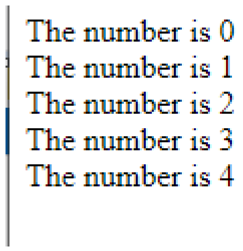
Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

Example:

```
body>

<div id="pp"></div>

 <script>

text=" ";

for (i=0; i<5;i++) {

  text += "The number is "+i+'</br>';


}

document.getElementById("pp").innerHTML=text;


</script>
```



```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

</body>From the example above, you can read:

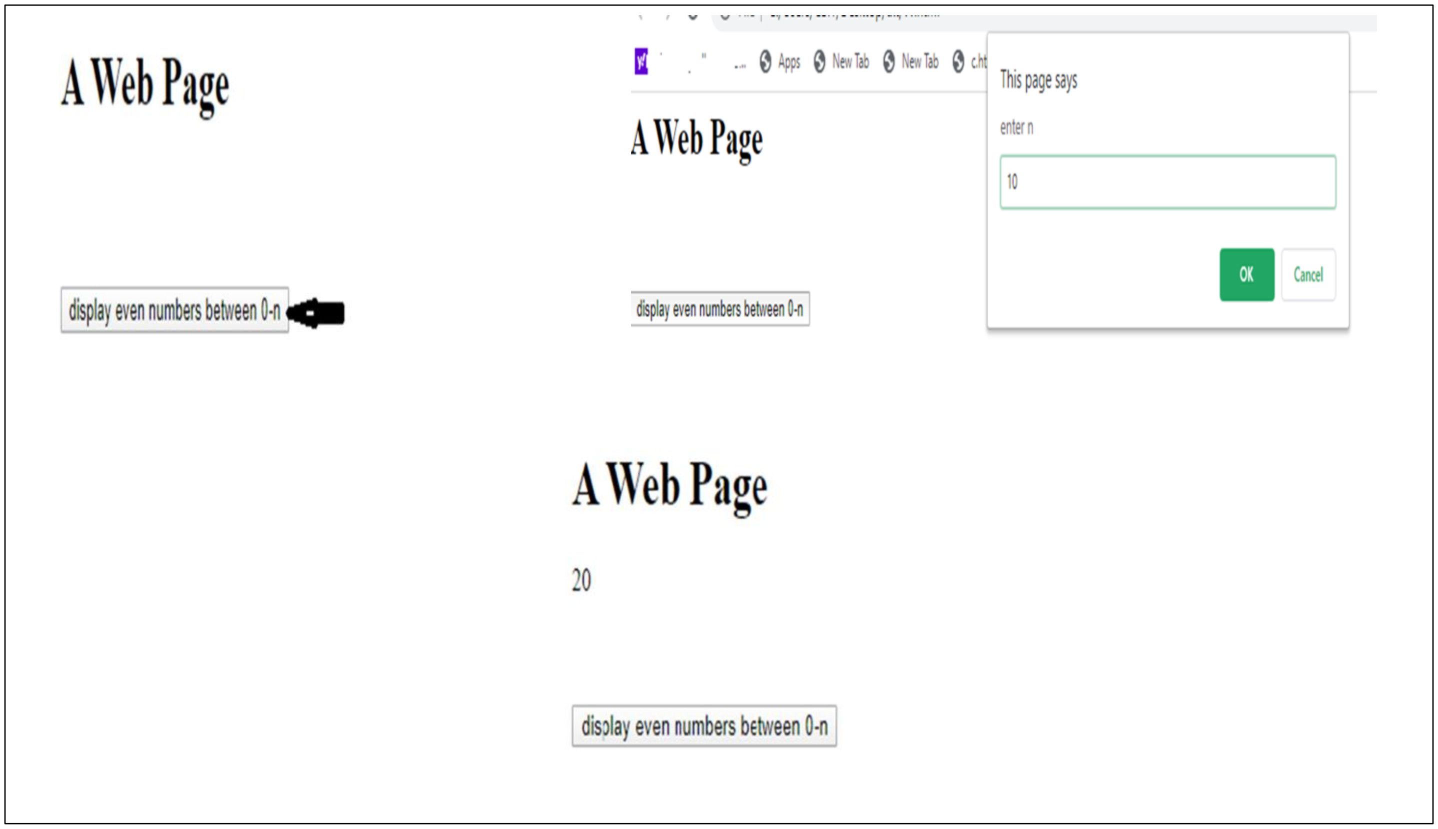
Statement 1 sets a variable before the loop starts (var i = 0).

Statement 2 defines the condition for the loop to run (i must be less than 5).

Statement 3 increases a value (i++) each time the code block in the loop has been executed.

```
<html>

<head>

<script>

function myFunction() {

n=prompt("enter n ");

x=0;

for(i=0;i<n;i=i+2)

{

  x=x+i;        جمع الاعداد الزوجيه

}

  document.getElementById("pp").innerHTML = x;

}

</script>
```

```
</head>

<body>

<h1>A Web Page</h1>

<div id="pp">    </div>          هذا هو العنصر

</br>  </br> </br>

<input  type="button"  value=" display even numbers between 0-n "
onclick="myFunction()"/>

</body>
```


A Web Page
display even numbers between 0-n
Apps
New Tab
New Tab
A Web Page
display even numbers between 0-n
OK
Cancel
A Web Page
display even numbers between 0-n

This page says

enter n

10

20

javaScript While Loop

Loops can execute a block of code as long as a specified condition is true.

The While Loop

The while loop loops through a block of code as long as a specified condition is true.

Syntax

```
while (condition) {
  // code block to be executed
}
```

Example

In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

Example

```
i=0;

while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

If you forget to increase the variable used in the condition, the loop will never end. This will crash your browser.

The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

Example

The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Example

```
i=0;

do {
  text += "The number is " + i;
  i++;
}
while (i < 10);
```

Do not forget to increase the variable used in the condition, otherwise the loop will never end!

Comparing For and While

If you have read the previous chapter, about the for loop, you will discover that a while loop is much the same as a for loop, with statement 1 and statement 3 omitted.

The loop in this example uses a for loop to collect the car names from the cars array:

Example

```
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";

for (;cars[i];) {      هذه غير مطلوبه forصيغه
  text += cars[i] + "<br>";
  i++;
}
```

The loop in this example uses a while loop to collect the car names from the cars array:

Example

```
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";

while (cars[i]) {
  text += cars[i] + "<br>";
  i++;
}
```

++