Welcome to BASIS-SEIP

WEB DESIGN

BASIS
Bangladesh Association of Software & Information Services

SEIP
Skills for Employment Investment Program

Sunday - Thursday

I’AM

Md Abdullah Al Noman

Trainer

Of

BASIS, BITM, PencilBox, NSDA, BTEB

01515 283053

abnoman.excel@gmail.com

Class Plan

1. HTML
2. CSS
3. Java Script [Fundamental]
4. CSS Framework
5. UI/UX to Web
6. Git & Github
7. PHP / Python
8. OOP
9. Laravel / Django
10. Final Project

Reference Website

1. https://w3schools.com/
2. https://webcoachbd.com/
3. https://getbootstrap.com/
4. https://youtube.com/
5. https://google.com/

Bootstrap

Essential Tools For Web Design & Dev

Editor/IDE [Integrated Development Environment]

- Notepad ++
- Sublime Text
- VS Code
- Atom
- Php Strom

Essential Tools For Web Design & dev

Browser / Server

- Mozilla Firefox
- Google Chrome
- Microsoft Edge

Editing Tools

- Adobe Photoshop/Canva

You can call to Web Design in many way

- Web Design
- Web Page Design
- Web Site Design
- Layout Design
- Landing Page Design
- Template Design
- User Interface (UI) Design
- Front End Design
- Front End Development

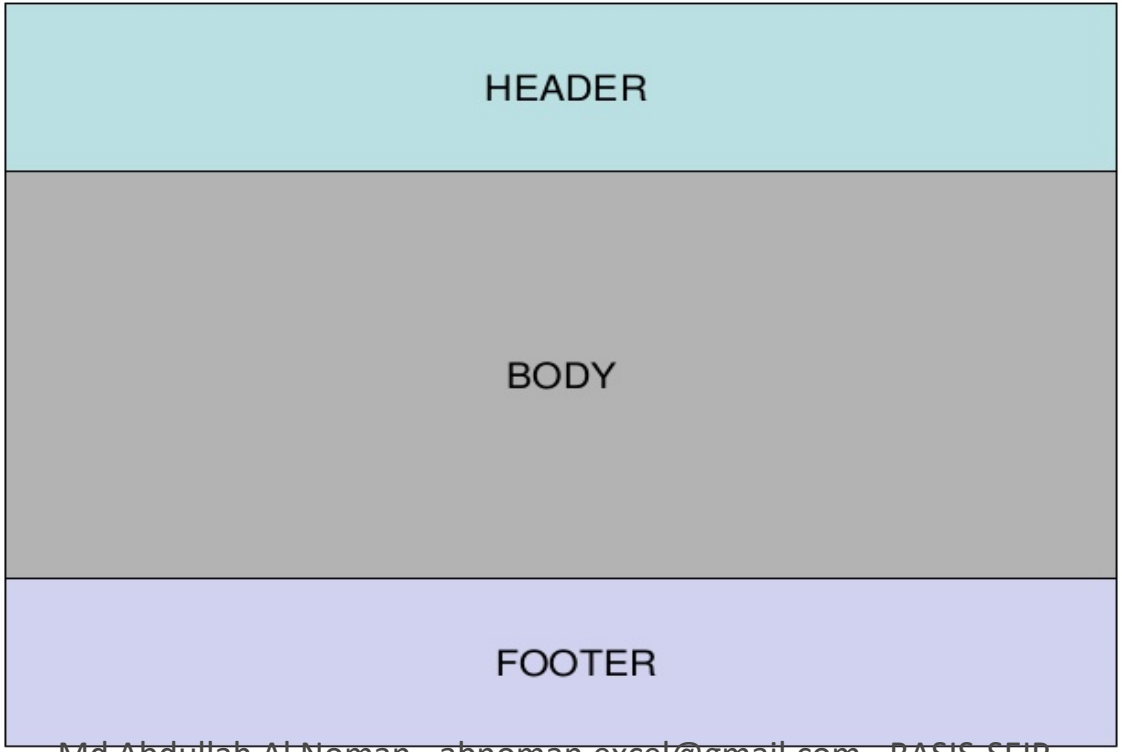
Guideline for Frontend Developer

- HTML
- CSS
- JavaScript (JS)
- CSS Framework - Bootstrap / Tailwind
- JS Framework - React JS + Next JS / Vue JS
- GIT System - Github / Gitlab
- Adobe Photoshop, Figma, Canva

Guideline for Frontend Developer

- You can Visit: https://tinyurl.com/web-seip

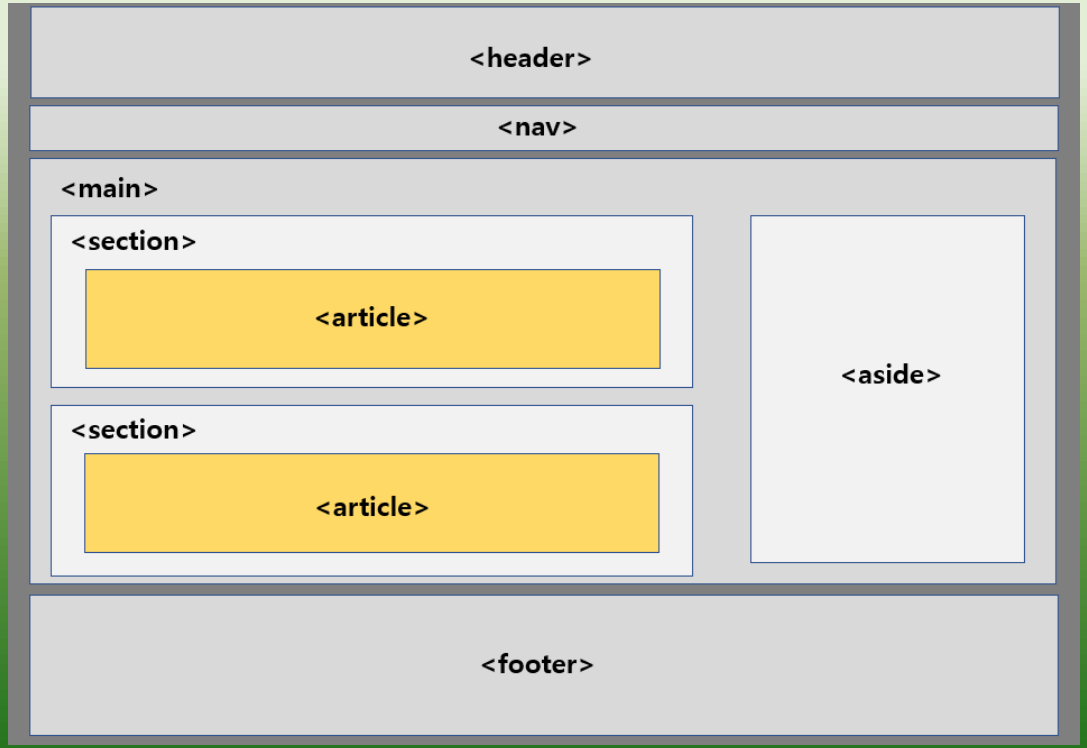
Web Page



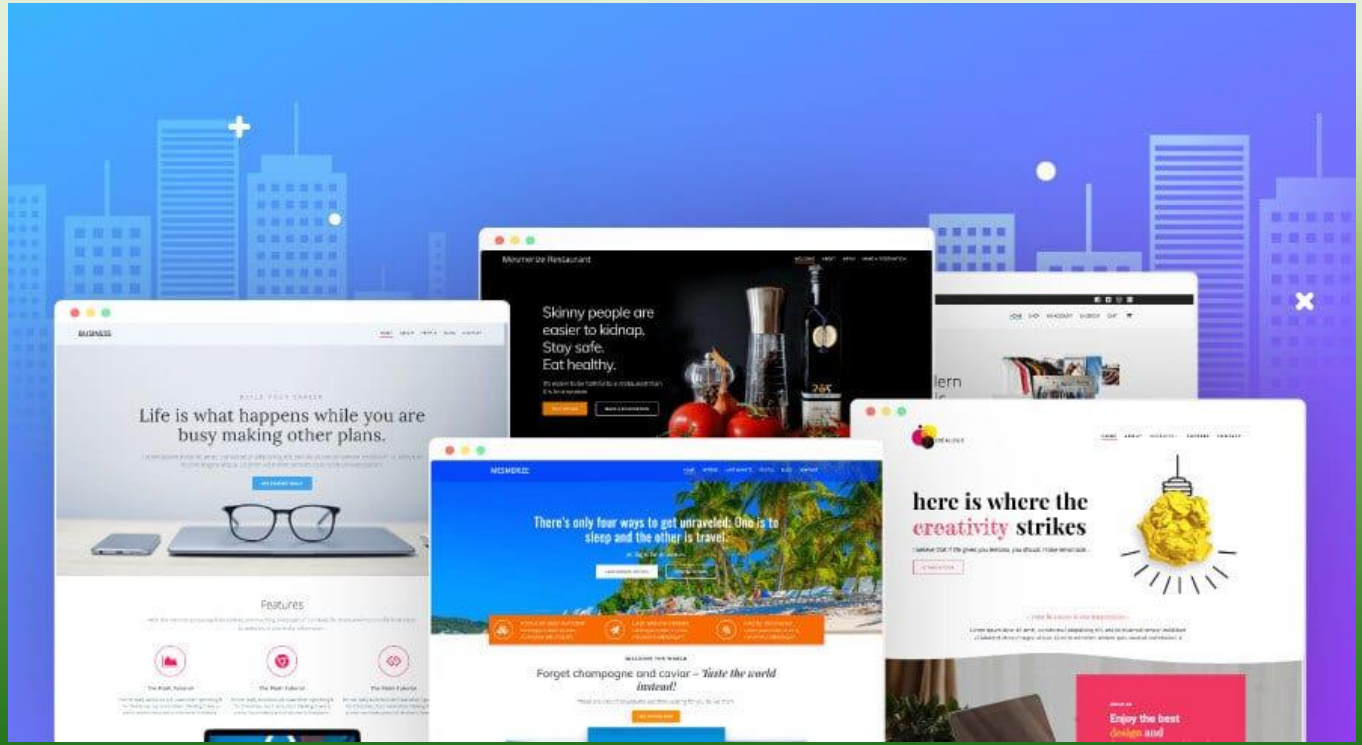
Web Page


Header
Navigation Menu
Content
Main Content
Content
Footer

Web Page

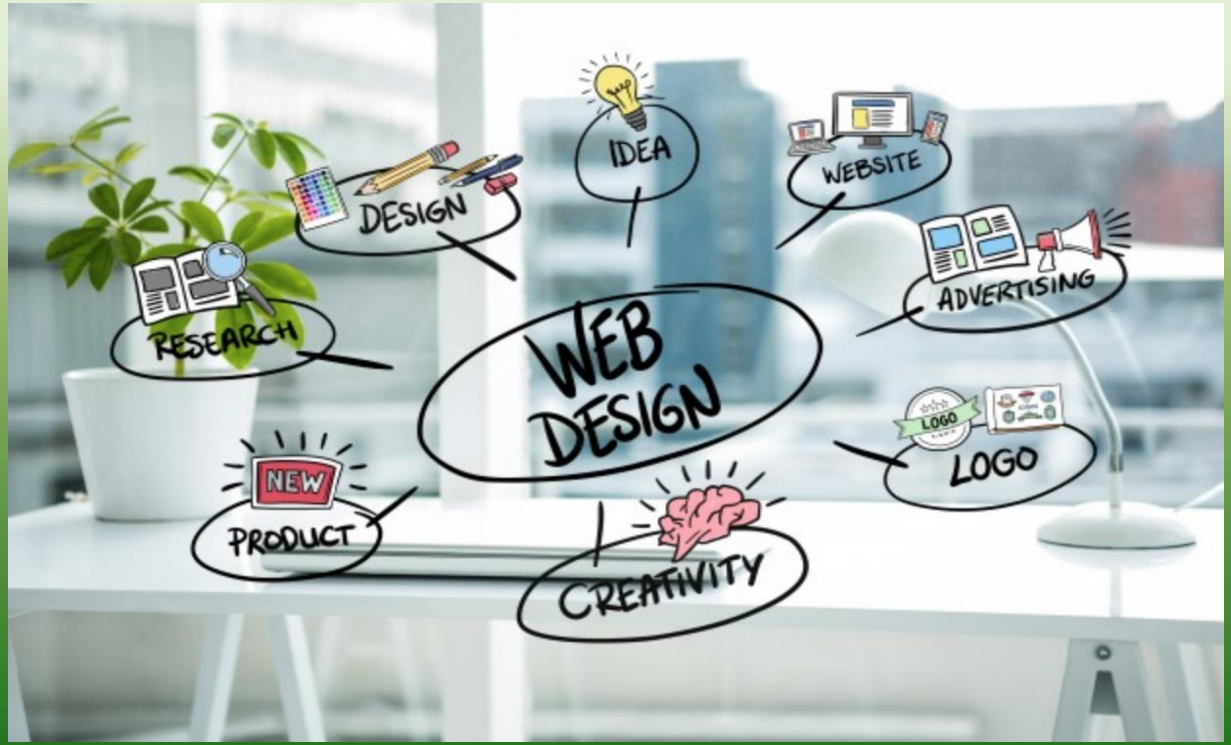

<header>
<nav>
<main>
<section>
<article>
<section>
<article>
<aside>
<footer>

Web Site


Skinny people are easier to kidnap. Stay safe. Eat healthy.
Life is what happens while you are busy making other plans.
Features
There's only four ways to get unraveled. One is to sleep and the other is travel.
Forget champagne and caviar – Taste the world instead!
here is where the creativity strikes
Enjoy the best

Web Design

Web Design


IDEA
WEBSITE
DESIGN
RESEARCH
WEB DESIGN
ADVERTISING
LOGO
LOGO
NEW
PRODUCT
CREATIVITY

Part of Web Designing

- Structure & Content

[HTML]

- Presentation

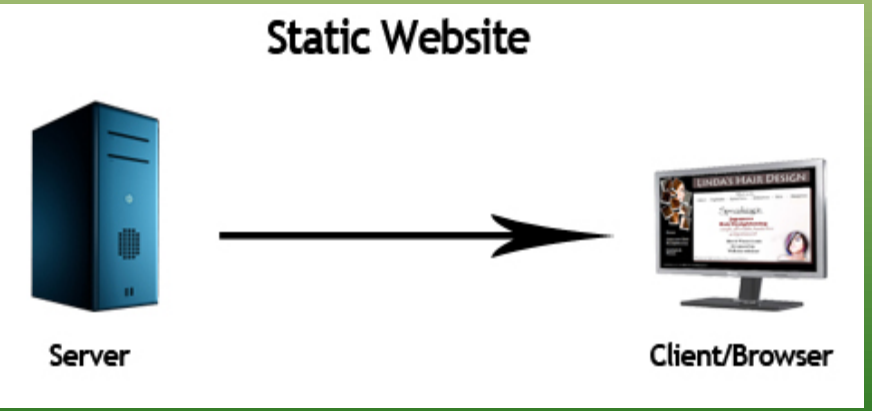
[Css, JavaScript]

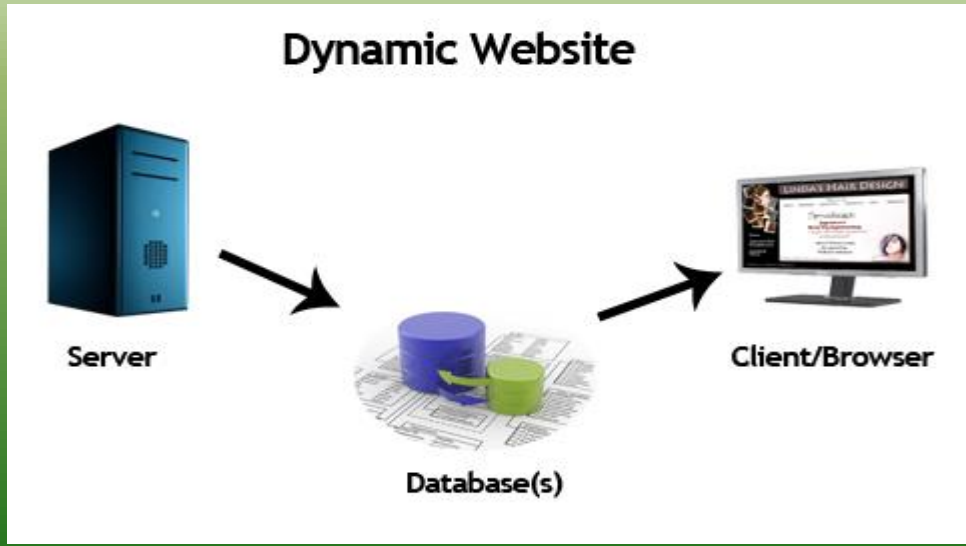
- Behavior

[Browser]

Type of Web Site

- Static Website
- Dynamic Website

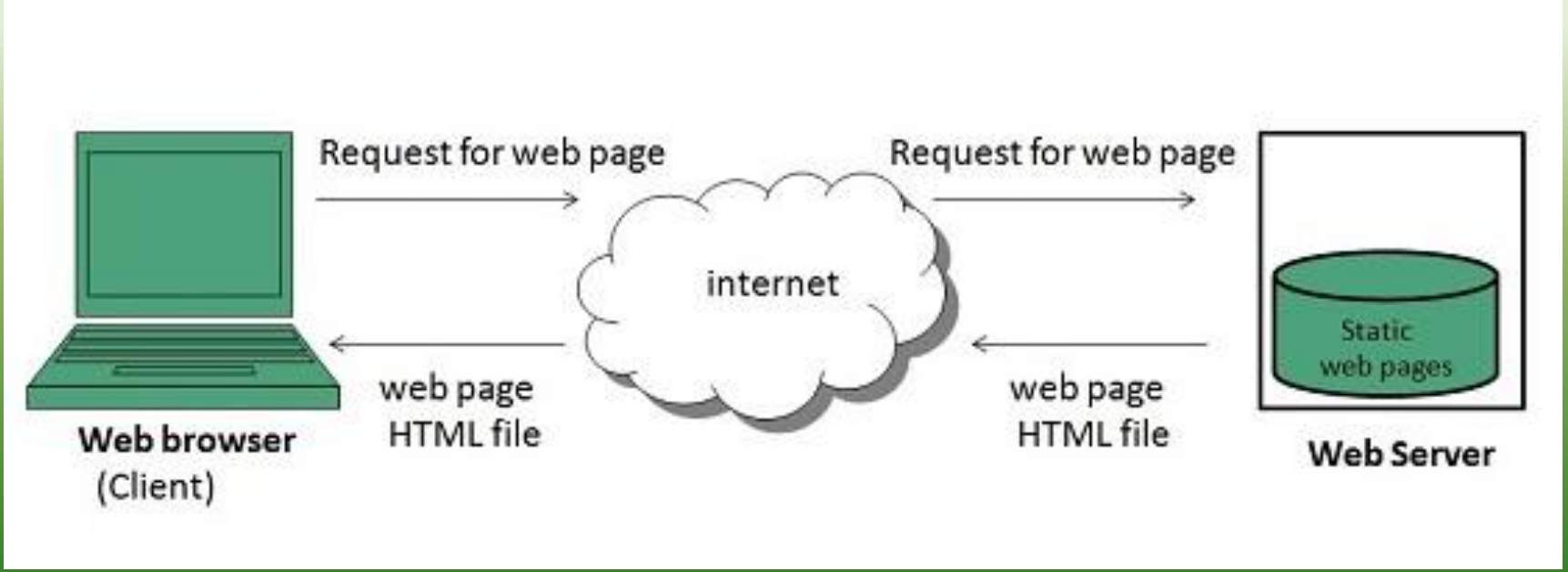




Static Web Site

A **static website** contains **Web pages** with fixed content. Each page is coded in plain HTML and displays the same information to every visitor. Unlike dynamic websites, they do not require any **Web** programming or database design.

Static Web Site



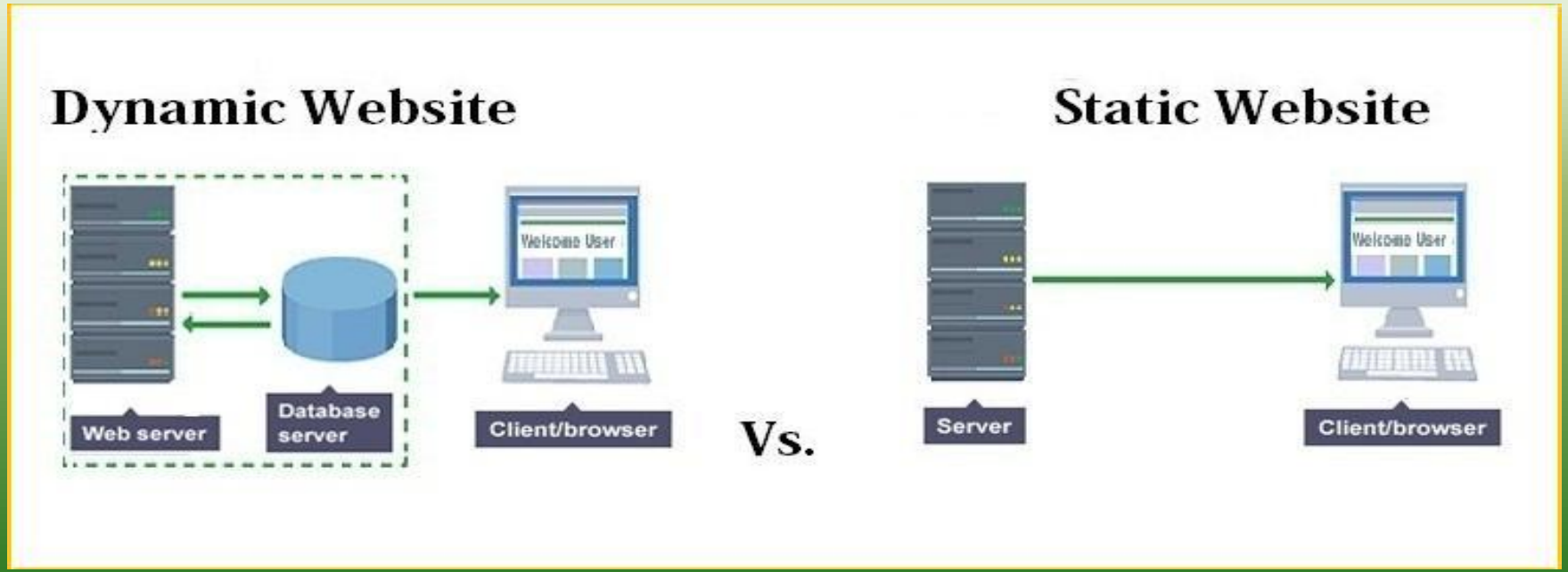
Dynamic Web Site

Dynamic website is a collection of dynamic web pages whose content changes dynamically. It accesses content from a database. Therefore, when you alter or update the content of the database, the content of the website is also altered or updated.

Dynamic website uses client-side scripting such as JS or server-side scripting such as PHP, or both to generate dynamic content.

It may generate separate HTML for each request based on user request.

Static and Dynamic Web Site

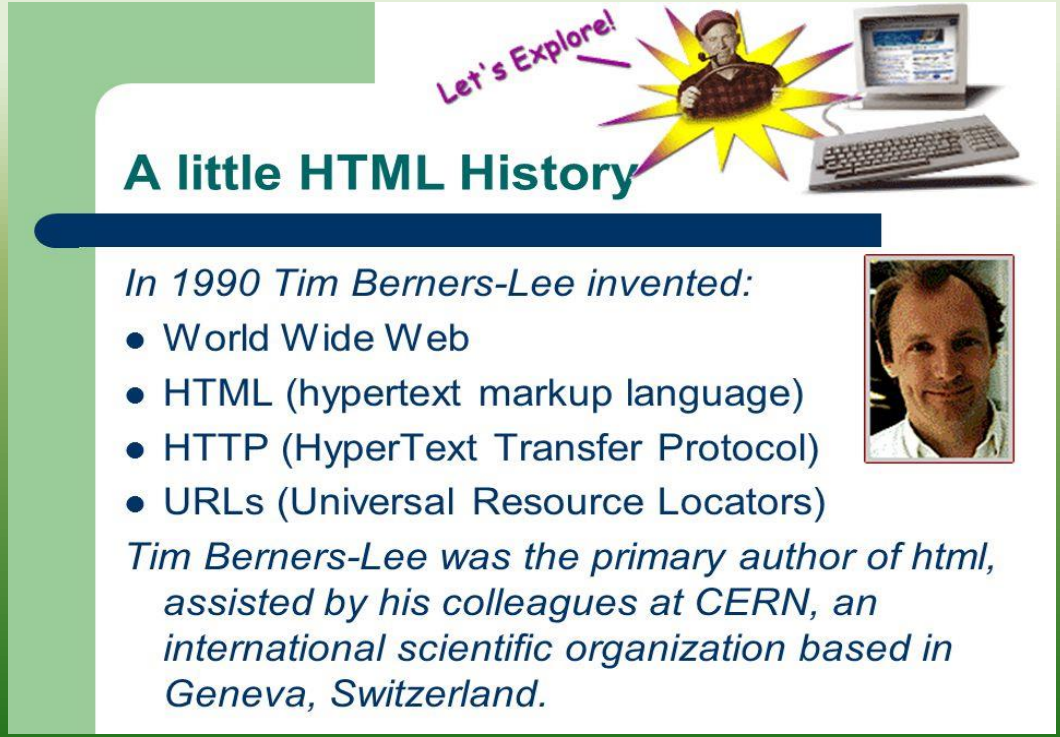


HTML(Hypertext Markup Language)

History Of HTML Version

1. HTML 1.0
2. HTML 2.0
3. HTML 3.0
4. HTML 3.2
5. HTML 4.01
6. XHTML 1.0
7. HTML5

History of HTML (bu.edu)



About HTML

Hypertext Markup Language

It's a **Tag** based **Markup Language**, who build up website Structure and Content.

Sir Tim Berner's Lee Invented HTML in 1989. First Developed HTML in 1990. First Version Of HTML released in 1991. Its Called HTML 1.0

HTML Tag

Hypertext Markup Language

Tag is a Syntax. It describes how to show anything in the browser.

Markup Language

All the tags are together called Markup Language

HTML Tag

<tag name >

<body>

HTML Tag

</tag name >

</body>

HTML Tag

< tag name > …. < / tag name >

** Hello Bangladesh **

How many HTML tags are there?

Total Number of HTML tags

Reference Website	Total number of HTML tags
Mozilla Developer Network(MDN)	142
HTML.com	132
W3schools.com	119
Eastmanreference.com	115
Htmlreference.io	113
Yourhtmlsource.com	69

How many HTML tags are there?

Current usable HTML tags

These HTML tags are part of the current version of HTML (HTML5).

Reference Website	Total number Supported HTML tags
Mozilla Developer Network(MDN)	111
Htmlreference.io	113
W3schools.com	107

HTML Element

An HTML element usually consists of a **start tag** and **end tag**, with the content inserted in between.

<starting tag> ….. </ending tag>

** Welcome to BASIS-SEIP **

Nested Element

When an element, inserted between another element it is called

Nested Elements.

```
<head>
    <title> Welcome </title>
</head>
```

HTML5 Structure

```
<!doctype html>
<html>
    <head>
        <meta charset=“UTF-8”>
        <title>Professional Web Design</title>
    </head>
    <body>
        <p> Welcome to BASIS-SEIP </p>
    </body>
</html>
```

Doctype of Others HTML

Version	Doctype
HTML5	<!doctype html>
HTML 4.01	<! doctype HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
XHTML 1.1	<! doctype html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

HTML Extension

Code Validation

You can validated your code using….

https://validator.w3.org/

HTML Standard

The **World Wide Web Consortium (W3C.org)**

W3C is the principle organization that sets standards for HTML

Heading Tag <h1> .. </h1>

- <h1> h1 Heading </h1>
- <h2> h2 Heading </h2>
- <h3> h3 Heading </h3>
- <h4> h4 Heading </h4>
- <h5> h5 Heading </h5>
- <h6> h6 Heading </h6>

Paragraph Tag ..

 Basis Institute of Technology and Management

Example of Paragraph Tag ..

```
<html>
<body>

<p>
    NetBeans Community Distributions are available under a Dual License consisting of the Common Development and
    Distribution License (CDDL) v1.0 and GNU General Public License (GPL) v2. Such distributions include additional
    components under separate licenses identified in the License file. See the Third Party License file for external
    components included in NetBeans and their associated licenses.
</p>

<p>
    NetBeans Community Distributions are available under a Dual License consisting of the Common Development and
    Distribution License (CDDL) v1.0 and GNU General Public License (GPL) v2. Such distributions include additional
    components under separate licenses identified in the License file. See the Third Party License file for external
    components included in NetBeans and their associated licenses.
</p>

</body>
</html>
```

Formatting Tag

-   - Bold text
-   - Important text
-   - Italic text
-   - Emphasized text
- <u> Underline Text </u>
- <mark>   - Marked text </mark>

Formatting Tag

- <del> - Deleted text </del>
- <ins> - Inserted text </ins>
- _{- Subscript text}
- ^{- Superscript text}
- <small> - Small text </small>
-
 - Inserts a single line break
- <hr/> - line in text

Pre - Formatting Tag

- **<pre>**
  - Defines pre-formatted text
- **</pre>**

HTML Quotation and Citation Elements

- Abbr - Defines an abbreviation or acronym
- Address - Defines contact information for the author/owner of a document
- Bdo - Defines the text direction
- Blockquote - Defines a section that is quoted from another source
- Cite - Defines the title of a work
- Q - Defines a short inline quotation

HTML <q> for Short Quotations

Browsers usually insert quotation marks around the <q> element.

Example:

```
<p> WWF's goal is to:
<q>Build a future where people live in harmony with nature.</q>
</p>
```

HTML < blockquote> for Short Quotations

Here is a quote from WWF's website:

<blockquote cite="http://www.worldwildlife.org/who/index.html">
For 50 years, WWF has been protecting the future of nature.
The world's leading conservation organization.
</blockquote>

HTML <abbr> for Abbreviations

Marking abbreviations can give useful information to browsers, translation systems and search-engines.

Example:

The **<abbr title="World Health Organization">WHO</abbr>** was founded in 1948.

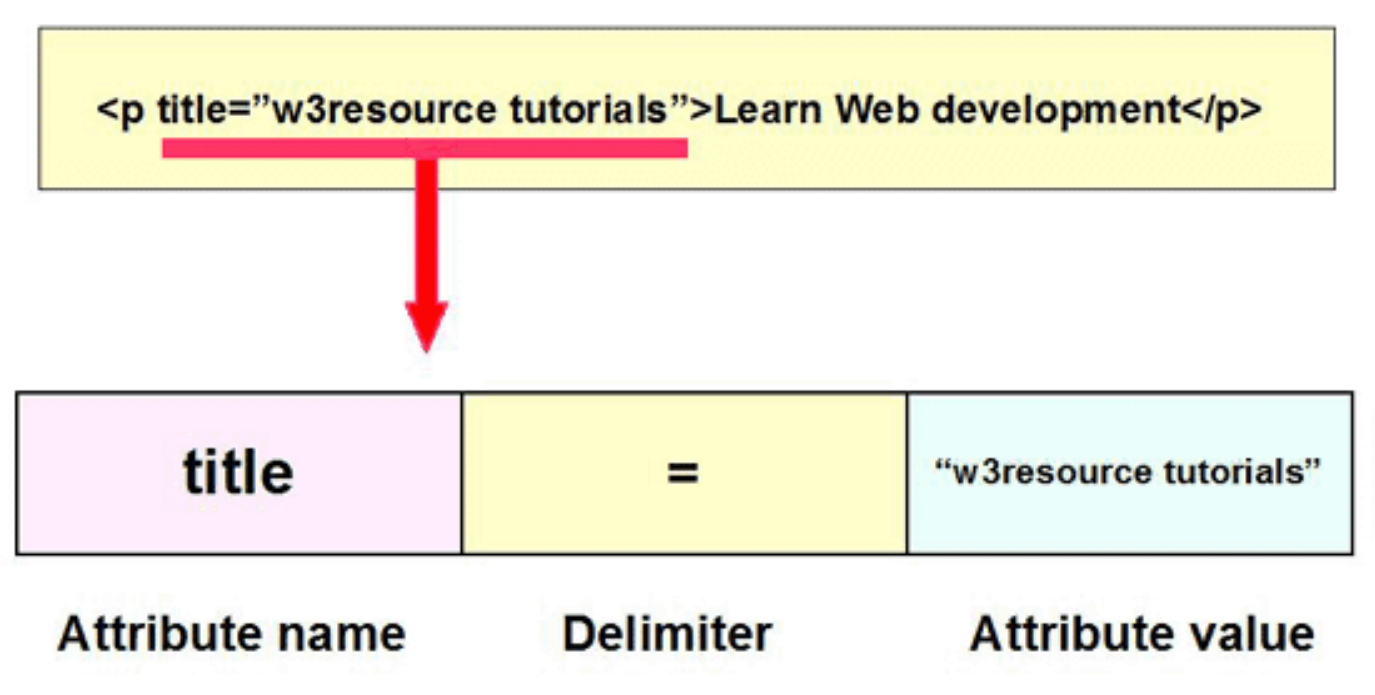
HTML <address> for Contact Information

```
<address>
        Written by John Doe.<br>
        Visit us at:<br>
        Example.com<br>
        Box 564, Disneyland<br>
        USA
</address>
```

Attribute

Attributes provide additional information about HTML elements.. In **HTML** syntax, an **attribute** is added to an **HTML** start tag.

Attribute


Learn Web development
title
=
"w3resource tutorials"
Attribute name
Delimiter
Attribute value

Attribute Condition

- Attribute name and values separated by an

  Equal [=] Sign

- Multiple Attribute separated by a **space**

HTML Comment Tags

<!-- Write your comments here -->

Link / Path

Link / Path 2 Types:

- **Absolute** Path/Link

  Absolute Path : **D:/BASIS/BITM/banner.jpg**

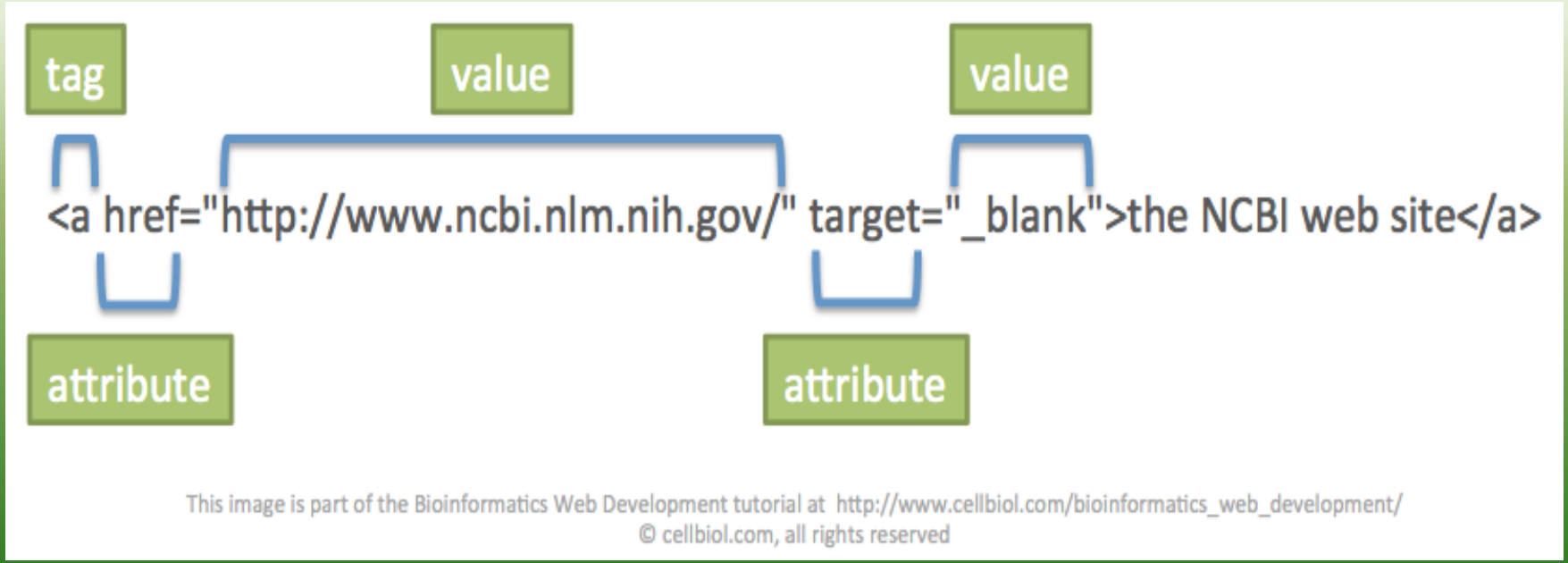
  Absolute Link : **http://bitm.org.bd/uploads/1416831456.jpg**

- **Relative** Path/Link

  Relative Path : **../bitm/banner.jpg**

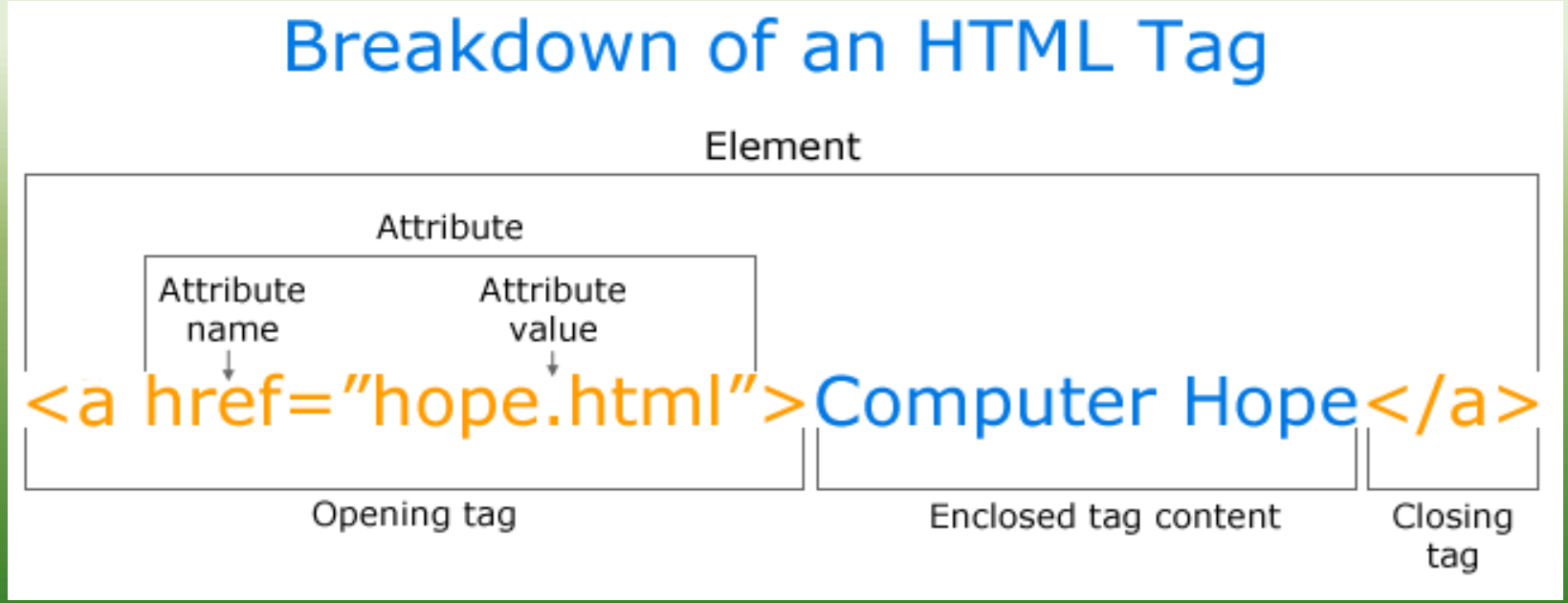
  Relative Link : XXXXXXXXXXXXX

A Tag



This image is part of the Bioinformatics Web Development tutorial at http://www.cellbiol.com/bioinformatics_web_development/

A Tag


Breakdown of an HTML Tag
Element
Attribute
Attribute name
Attribute value
Computer Hope
Opening tag
Enclosed tag content
Closing tag

Image

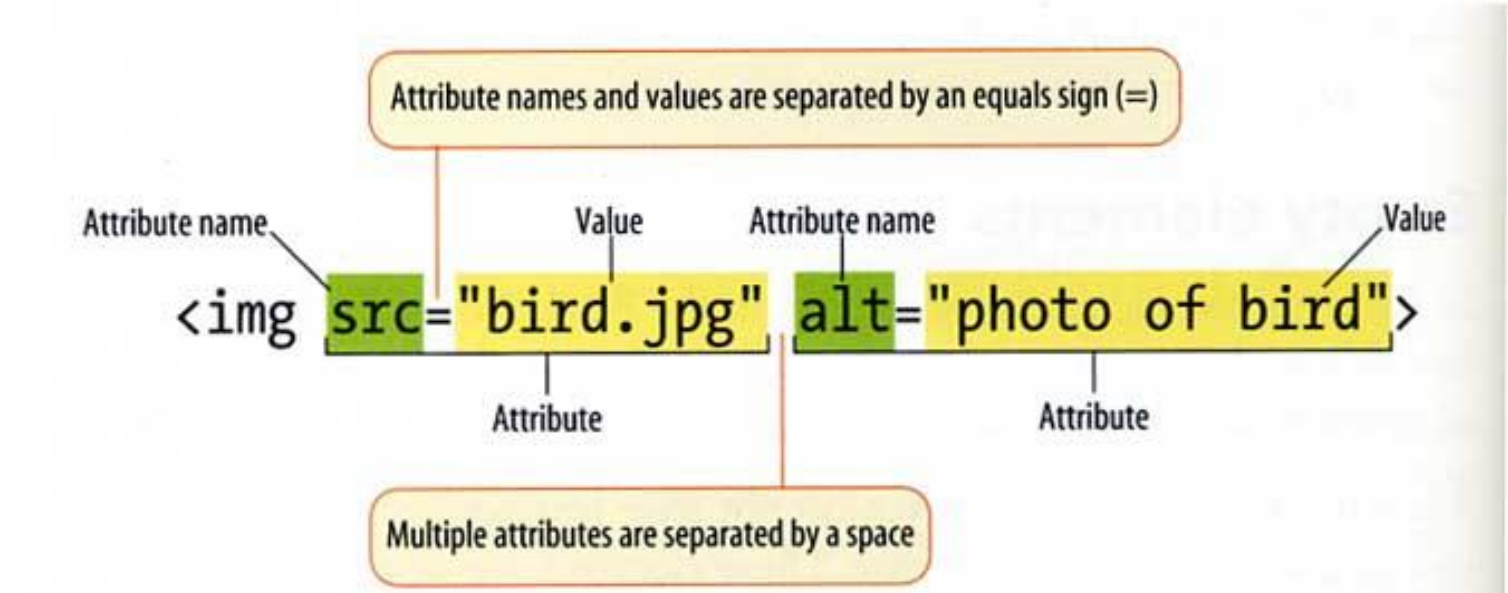


Figure 4-11. An element with attributes.

Table

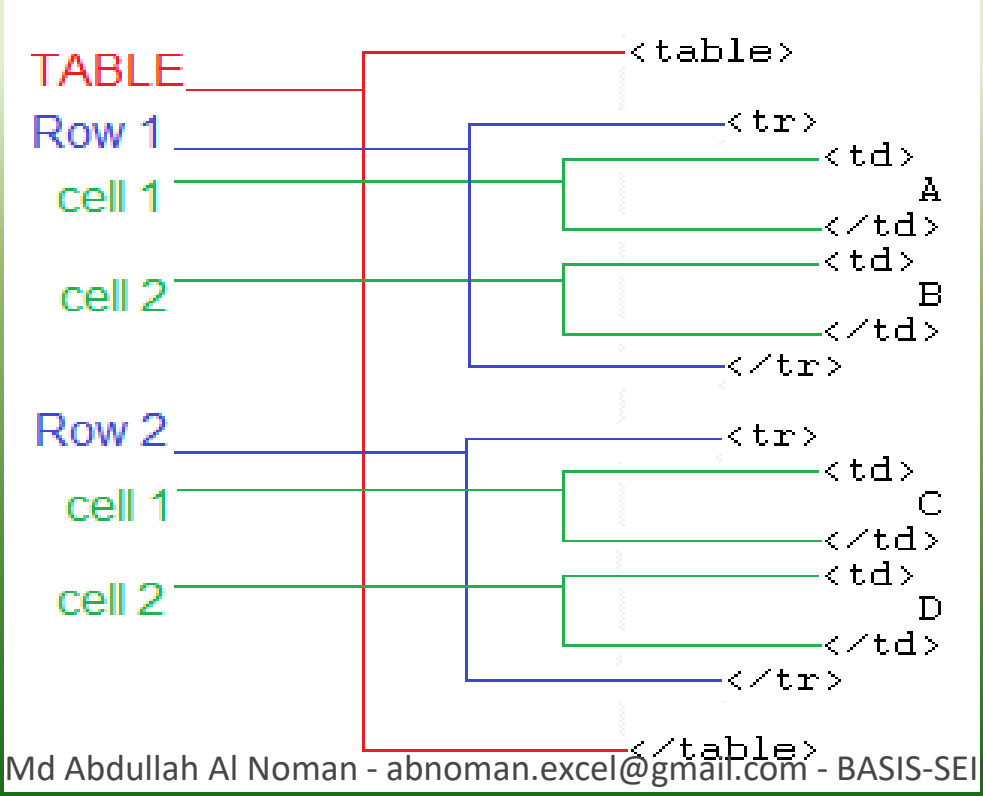


TABLE
Row 1
cell 1
cell 2
Row 2
cell 1
cell 2
<table>
<tr>
<td>
A
</td>
<td>
B
</td>
</tr>
<tr>
<td>
C
</td>
<td>
D
</td>
</tr>
</table>

Table Colspan

colspan

```
<table border="1" width="30%">
<tr>
        <td>1</td>
        <td>2</td>
        <td>3</td>
</tr>
<tr>
        <td>4</td>
        <td>5</td>
        <td>6</td>
</tr>
<tr>
        <td>7</td>
        <td colspan="2">8</td>
</tr>
</table>
```

1	2	3
4	5	6
7	8	

????

Table Rowspan

rowspan

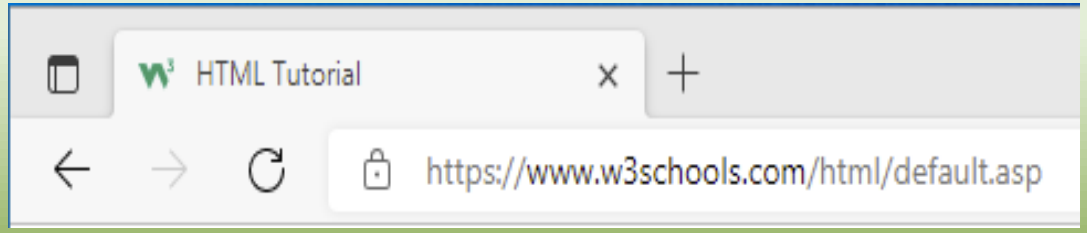
```
<table border="1" width="30%">
<tr>
        <td rowspan="2">1</td>
         <td>2</td>
         <td>3</td>
</tr>
<tr>
        ????
         <td>5</td>
         <td>6</td>
</tr>
<tr>
         <td>7</td>
         <td>8</td>
         <td>9</td>
</tr>
</table>
```

1	2	3
	5	6
7	8	9

Some Sequence Format

- Tag -> attribute = “attribute value ”
- Tag -> attribute -> properties =“Properties value ”
- Table -> tr -> td/th
- Tr = Table Row
- Td = Table Data
- Th = Table Heading
- Td / th / cell all are same

Favicon




```
<head>
  <title> Page Title</title>
  <link rel="icon" href="assets/image/favicon.ico">
</head>
```

AUDIO

```
<audio controls autoplay loop>
    <source src="horse.ogg" type="audio/ogg">
</audio>
```

VIDEO

```
<video width="320" height="240" controls>
    <source src="movie.mp4" type="video/mp4">
</video>
```

LIST UL /OL /LI

Unordered HTML List

Uses:

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

LIST UL /OL /LI

Unordered HTML List:

LIST Style Type : disc, circle, square, none

Uses:

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

LIST UL /OL /LI

Ordered HTML List:

Uses:

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

LIST UL /OL /LI

Ordered HTML List:

Order LIST Type : type="1“ type="A“ type="a“ type="I“ type="i"

Uses:

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

Description LIST DL /DT/DD

Uses:

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

1. **<dl>** element to define a description list
2. **<dt>** element to define the description term
3. **<dd>** element to describe the term in a description list

FORM

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

Syntax:

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

FORM

The <input> Element :

<input type="text"> Defines a one-line text input field

<input type=“number"> For Integer Value

<input type=“password"> For Secret Data Input

<input type=“email"> For Email

FORM

The <input> Element :

<input type="radio"> Defines a radio button (for selecting one of many choices)

<input type="checkbox"> defines a **checkbox**.

<input type="**button**" **onclick**="alert('Hello World!')" **value**="Click Me!"> defines a **Button**

FORM

The <input> Element :

<input type="submit"> Defines a submit button (for submitting the form)

Go on:
https://www.w3schools.com/html/html_forms.asp

FORM

Example 1:

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

FORM

Example 2:

```
<form>
 <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>
```

FORM

Example 2: Radio Button

```
<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>
```

FORM

Example 2: Checkbox

```
<form action="" method="">
  <input type="checkbox" name="vehicle" value="Bike">
 I have a bike<br>
  <input type="checkbox" name="vehicle" value="Car" checked> I
    have a car<br>
  <input type="submit" value="Submit">
</form>
```

FORM

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

```
<select name=“Web Design">
  <option value=“html">HTML</option>
  <option value=“css">CSS</option>
</select>
```

FORM

Optgroup : Group related options with <optgroup> tags:

```
<select>
      <optgroup label="Swedish Cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
      </optgroup>
      <optgroup label="German Cars">
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
      </optgroup>
    </select>
```

FORM

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

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

FORM

A form with a keygen field:

```
<form>
    Username: <input type="text" name="user">
    Encryption: <keygen name="security">
    <input type="submit">
</form>
```

FORM

```
 <input list="browsers" name="browser">
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
<input type="submit">
```

FORM

Name	Value
Name	
Sex	○ Male ◉ Female
Eye color	green
Check all that apply	☐ Over 6 feet tall ☐ Over 200 pounds
Describe your athletic ability:	
Enter my information	

FORM


Absence Request Form
Name:
E-mail Address:
Position
Department:
Select Department
Reason For Absence?
Floating Holiday
Vacation
Sick Leave
FMLA
Other:
Dates of Absence:
mm/dd/yyyy
to
mm/dd/yyyy
Total Days Absent:
With Pay
Without Pay
Comments:

IFRAME

An inline frame is used to embed another document within the current HTML document.

<iframe src=“url" name=""></iframe>

W3Schools.com

Competed Tag List

html, head, title, meta, body, p, h1, h2, h3, h4, h5, h6, b, strong, I, em, u, ins, mark, sub, sup, small, hr, br, del, pre, table, tr, td, th, thead, tbody, tfoot, a, img, fieldset,leged, form,input,select,option,optgroup,textarebutton,datalist, ul, ol, li, dd, dl, dt, iframe,audio,video,source.

Competed Attribute List

charset, name, content,

border, height, width, colspan,rowspan

,src,alt,title,valign,align,

action,method,cellpadding,type,name,value,hidden,

placeholder,maxlength,readonly,min,max,requered

,accept,label,rows,cols,autocomplete, autofocus, list,disabled,Multiple,Controls,autoplay,loop.

CSS (Cascading Style Sheets)

Selector based Style sheet language

CSS (Cascading Style Sheets)

Cascading Style Sheets (**CSS**) is a stylesheet language used to describe the presentation of a document written in HTML .CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS (Cascading Style Sheets)

The CSS starts in 1994.

By Håkon Wium Lie. He was worked at CERN.

CSS (Cascading Style Sheets)

SYNTAX

Selector {

Property : value;

}

CSS (Cascading Style Sheets)

SELECTOR

Selector represent by many ways.

Basically Its might be 2 way:

1. Html tag name

2. User define name

CSS (Cascading Style Sheets)

How to call & How to define A Selector??

ID = #

CLASS = .

CSS (Cascading Style Sheets)

CSS CLASS APPLY IN HTML BY3 WAYS

Inline CSS

Internal CSS

External CSS

CSS (Cascading Style Sheets)

INLINE CSS

attribute=“properties : value;”

<p style=“color : black;”> HELLO </p>

CSS (Cascading Style Sheets)

INTERNAL CSS

```
<style>
{CSS Selector Here}
</style>
```

CSS (Cascading Style Sheets)

EXTERNAL CSS

File Extension .CSS

CSS (Cascading Style Sheets)

EXTERNAL CSS LINK

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

CSS (Cascading Style Sheets)

COLOR

3 way to use color in CSS:

1. a valid color name - like "red"
2. an RGB value - like "rgb(255, 0, 0)"
3. a HEX/HTML value - like "#ff0000"

CSS (Cascading Style Sheets)

BACKGROUND

CSS background properties:

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

CSS (Cascading Style Sheets)

BACKGROUND

Background-Color: 3 way to use

1. Background-color: red;
2. Background-color: rgb(255,0,0);
3. Background-color: #c2c2c2;

CSS (Cascading Style Sheets)

BACKGROUND

Background-Image:

background-image: url(Image name or image link);

Ex: background-image: url(bitm.jpg);

CSS (Cascading Style Sheets)

BACKGROUND

Background-Image: Problem & Solution

background: url(bitm.jpg);

background-size: Width Height;

ex: background-size: 100% 100%;

background-position: center;

background-repeat: no-repeat;

CSS (Cascading Style Sheets)

BACKGROUND

Background-Image: Repeat

```
background-image: url(“bitm.jpg");

background-repeat: repeat-x or y;
```

CSS (BOX MODEL)

CSS (Cascading Style Sheets)

BORDER

CSS Border: 3 way to use

1. border: 3px solid red;
2. border: 3px solid #e2e2e2;
3. border: 3px solid rgb(255,0,0);

CSS (Cascading Style Sheets)

BORDER

Border Style

dotted ,dashed ,solid ,double ,groove ,ridge ,inset ,outset ,none ,hidden

CSS (Cascading Style Sheets)

BORDER

Border Style Use

```
border-style: dotted ;
border-color: green;
border-width: 5px;
```

CSS (Cascading Style Sheets)

BORDER

Border Style Use

```
border-top-style: dotted;
border-right-style: solid;
border-bottom-style: dotted;
border-left-style: solid;
```

CSS (Cascading Style Sheets)

MARGIN

Margin – Short hand Property

```
margin-top: 100px;
margin-bottom: 100px;
margin-right: 100px;
margin-left: 100px;
```

CSS (Cascading Style Sheets)

MARGIN

Margin – Property have 4 value

margin: top right bottom left;

Ex: margin: 10px 10px 10px 10px;

Total margin ex: margin:10px;

CSS (Cascading Style Sheets)

PADDING

padding – Short hand Property

```
padding-top: 100px;
padding-bottom: 100px;
padding-right: 100px;
padding-left: 100px;
```

CSS (Cascading Style Sheets)

PADDING

Padding– Property have 4 value

padding: top right bottom left;

Ex: padding : 10px 10px 10px 10px;

Total padding ex: padding:10px;

CSS (Cascading Style Sheets)

TYPOGRAPHY

Typography is the art and technique of arranging type to make written language legible, readable and appealing when displayed. The arrangement of type involves selecting typefaces, point sizes, line lengths, line-spacing, and letter-spacing, as well as adjusting the space between pairs of letters.

CSS (Cascading Style Sheets)

TEXT FORMATTING | TYPOGRAPHY

Text color-> color: **blue**;

Text Alignment-> text-align: **center | left| right |justify**;

Text-Decoration-> text-decoration: **none | overline |line-through | underline;**

Text Transformation-> text-transform: **uppercase | lowercase | capitalize;**

CSS (Cascading Style Sheets)

TEXT FORMATTING

Text Indentation-> **text-indent: 50px;**

Letter Spacing-> **letter-spacing: 3px;**

Word Spacing-> **word-spacing: 10px;**

text Shadow-> **text-shadow: 3px 2px 3px red;**

Line Height-> **line-height: 1.5;**

CSS (Cascading Style Sheets)

FONTS

```
@font-face{
   src: url("URL/SOURCE");
   font-family: myFont;
}
```

EX: use in p:

```
p{
      font-family: myFontName;
   }
```

CSS (Cascading Style Sheets)

FONTS

font-size: medium|xx-small|x-small|small|large|x-large|xx-large|smaller|larger|length|initial|inherit;

font-style: normal|italic|oblique|initial|inherit;

font-weight: normal|bold|bolder|lighter|number|initial|inherit;

CSS (Cascading Style Sheets)

ICONS

1. Font Awesome Icons
2. Bootstrap Icons
3. Remix Icons

CSS (Cascading Style Sheets)

LIST

UL/OL->LI:

list-style-type: circle|square|upper-roman|lower-alpha;

list-style-image: url(image link);

CSS (Cascading Style Sheets)

ELEMENT

Block Element

Inline Element

CSS (Cascading Style Sheets)

DISPLAY

Display Property depend on 2 Element type:

1. BLOCK ELEMENT.
2. INLINE ELEMENET.

use Ex: display: block | inline | none;

CSS (Position)

- The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute, sticky).
- Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

CSS (Position)

There are four different position values:

1. **Static**
2. **Relative**
3. **Fixed**
4. **Absolute**

CSS (Position)

Position: Static

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.

CSS (Position)

Position: Static Example:

```
<head>
<style>
div.static {
    position: static;
    border: 3px solid #73AD21;
}
</style>
</head>
```

P.T.O

CSS (Position)

Position: Static Example:

```
<body>
<h2>position: static;</h2>
<p>An element with position: static; is not positioned in any special way; it is
always positioned according to the normal flow of the page:</p>

<div class="static">
This div element has position: static;
</div>
</body>
```

CSS (Position)

Position: Relative:

If you set position: relative; on an element but no other positioning attributes (top, left, bottom or right), it will no effect on it's positioning at all, it will be exactly as it would be if you left it as position: static; But if you do give it some other positioning attribute, say, top: 10px;, it will shift its position 10 pixels down from where it would normally be.

CSS (Cascading Style Sheets)

Position: Relative Example:

```
<style>
div.relative {
    position: relative;
    left: 30px;
    border: 3px solid #73AD21; }
</style>
```

CSS (Cascading Style Sheets)

Position: Relative Example:

```
</head>
<body>
<h2>position: relative;</h2>
<p>An element with position: relative; is positioned relative to its normal
    position:</p>
<div class="relative">
This div element has position: relative;
</div>
```

CSS (Cascading Style Sheets)

Position: Fixed:

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
- A fixed element does not leave a gap in the page where it would normally have been located.

CSS (Cascading Style Sheets)

Position: Fixed example:

```
<style>
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;}
</style>
```

CSS (Cascading Style Sheets)

Position: Fixed example:

```
<body>
<h2>position: fixed;</h2>
<p>An element with position: fixed; is positioned relative to the viewport, which means it
    always stays in the same place even if the page is scrolled:</p>
<div class="fixed">
This div element has position: fixed;
</div>
</body>
```

CSS (Cascading Style Sheets)

Position: Absolute:

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

CSS (Cascading Style Sheets)

Position: Absolute Example:

```
<style>
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
} 
```

CSS (Cascading Style Sheets)

Position: Absolute Example:

```
div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21; } </style>
```

CSS (Cascading Style Sheets)

Position: Absolute Example:

```
<body>
<h2>position: absolute;</h2>
<p>An element with position: absolute; is positioned relative to the nearest
    positioned ancestor (instead of positioned relative to the viewport, like
    fixed):</p>
<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position: absolute;</div>
</div>
</body>
```

CSS (Cascading Style Sheets)

POSITION

Overlapping:

- When elements are positioned, they can overlap other elements.
- The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
- An element can have a positive or negative stack order.

CSS (Cascading Style Sheets)

Overlapping Example:

```
<style>
img {
    position: absolute;
    left: 0px;
    top: 0px;
    z-index: -1;
}
</style>
```

CSS (Cascading Style Sheets)

Overlapping Example:

```
<body>


<h1>This is a heading</h1>
<img src="w3css.gif" width="100" height="140">
<p>Because the image has a z-index of -1, it will be placed behind the text.</p>


</body>
```

CSS (Cascading Style Sheets)

OVERFLOW

The CSS overflow property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

CSS (Cascading Style Sheets)

OVERFLOW

1. visible - Default. The overflow is not clipped. It renders outside the element's box
2. hidden - The overflow is clipped, and the rest of the content will be invisible
3. scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content
4. auto - If overflow is clipped, a scrollbar should be added to see the rest of the content
5. **Note:** The overflow property only works for block elements.

CSS (Cascading Style Sheets)

CLEARFIX

1. If an element is taller than the element containing it, and it is floated, it will overflow outside of its container.
2. Then we can add overflow: auto; to the containing element to fix this problem.
3. The overflow:auto clearfix works well as long as you are able to keep control of your margins and padding (else you might see scrollbars). The **new, modern clearfix hack** however, is safer to use, and the following code is used for most webpages:

CSS (Cascading Style Sheets)

PSEUDO CLASS

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

Style an element when a user mouses over it

Style visited and unvisited links differently

Style an element when it gets focus

CSS (Cascading Style Sheets)

PSEUDO ELEMENS

A CSS pseudo-element is used to style specified parts of an element.

Ex:

1. Style the first letter, or line, of an element
2. Insert content before, or after, the content of an element
3. Selector :: pseudo-element {
   property:value;
   }

CSS (Cascading Style Sheets)

NAVIGATION [NAV]

Navigation Bar = List of Links

A navigation bar is basically a list of links, using the <ul> and <li> elements makes perfect sense.

CSS (Cascading Style Sheets)

CSS OPACITY / TRANSPARENCY

The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent:

Ex:
```
opacity: 0.5;
    filter: alpha(opacity=50); /* For IE8 and earlier */
```

CSS (Cascading Style Sheets)

TOOLTIP

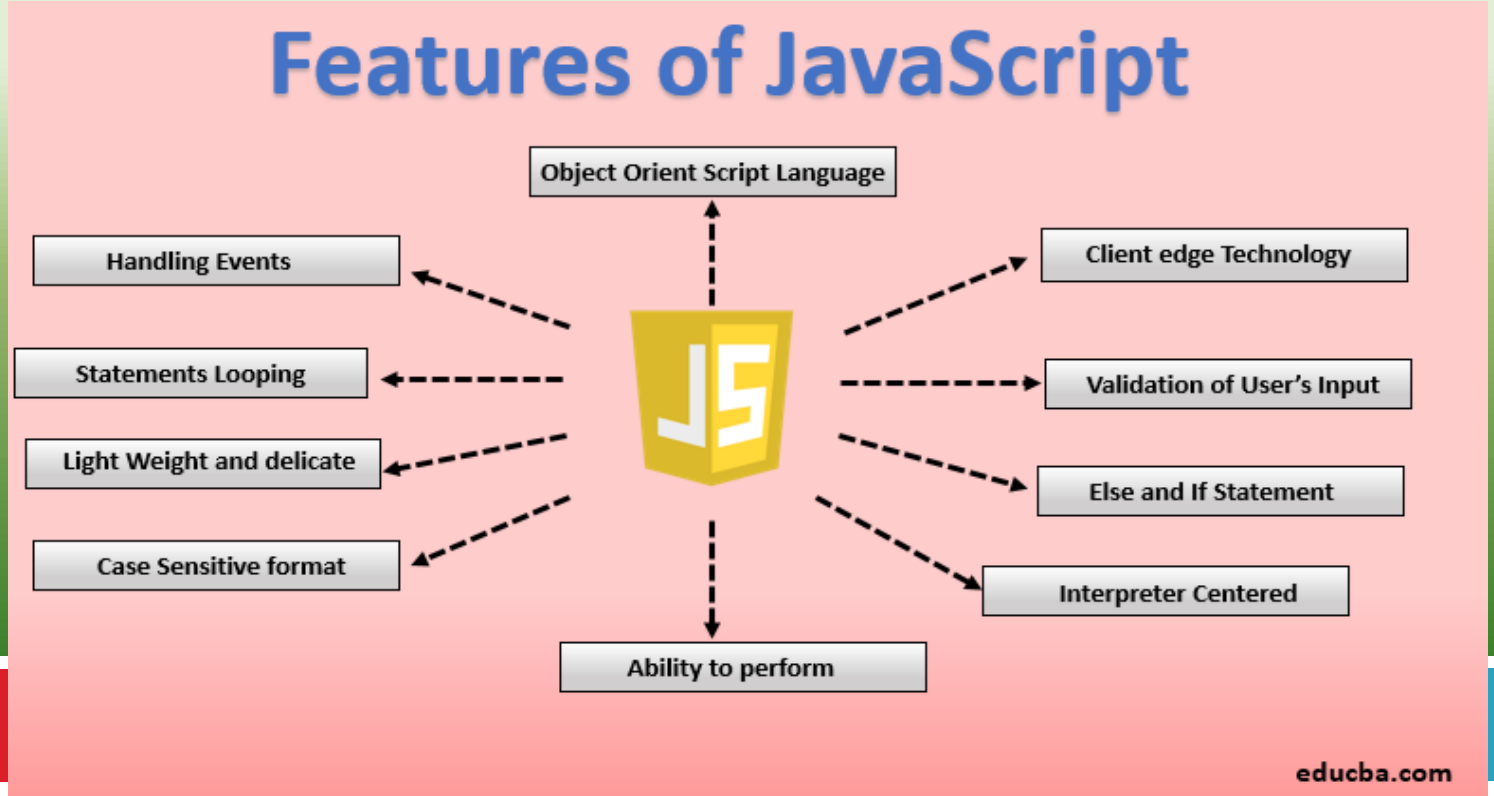
A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element.

CSS (Cascading Style Sheets)

IS END??

NEVER: JUST GO ON -> HTTPS://WWW.W3SCHOOLS.COM/CSS/DEFAULT.ASP

JAVASCRIPT


Features of JavaScript
Object Orient Script Language
Handling Events
Client edge Technology
Statements Looping
Validation of User's Input
Light Weight and delicate
Else and If Statement
Case Sensitive format
Interpreter Centered
Ability to perform
educba.com

JAVASCRIPT

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

JavaScript ?

- JavaScript is the world's most popular programming language.
- JavaScript is the programming language of the Web.
- JavaScript is easy to learn.

Did You Know?

- JavaScript and Java are completely different languages, both in concept and design.
- JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.
- ECMA-262 is the official name of the standard. ECMAScript is the official name of the language.

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

Version of JavaScript (ref:W3schools)

- The Original JavaScript ES1 ES2 ES3 (1997-1999)
- The First Main Revision ES5 (2009)
- The Second Revision ES6 (2015)
- The Yearly Additions (2016, 2017, 2018)

Commonly Asked Questions ?????

- How do I get JavaScript?
- Where can I download JavaScript?
- Is JavaScript Free?
- ANS:
  - **You don't have to get or download JavaScript.**
  - **JavaScript is already running in your browser on your computer, on your tablet, and on your smart-phone.**
  - **JavaScript is free to use for everyone.**

JavaScript Where To

- In HTML, JavaScript code is inserted between <script> and </script> tags.
- JavaScript in <head>
- JavaScript in <body>
- External JavaScript

External References

- An external script can be referenced in 3 different ways:
- With a full URL (a full web address)
- With a file path (like /js/)
- Without any path
- <script src="https://www.w3schools.com/js/myScript.js"></script>

JavaScript Can Change HTML Content

- JavaScript Can Change HTML Attribute Values
  - document.getElementById("demo").innerHTML = "Hello JavaScript";
- JavaScript Can Change HTML Styles (CSS)
  - document.getElementById("demo").style.fontSize = "35px";
- JavaScript Can Hide HTML Elements
  - document.getElementById("demo").style.display = "none";
- JavaScript Can Show HTML Elements
  - document.getElementById("demo").style.display = "block";

JavaScript Output

JavaScript Display Possibilities

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

Remember that JavaScript identifiers (names) must begin with:

- A letter (A-Z or a-z)
- A dollar sign ($)
- Or an underscore (_)

Using innerHTML


```
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
```

Using document.write()

- <body>
- <script>
  document.write(5 + 6);
  </script>
- OR

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

</body>
```

Using window.alert()


```
<body>

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

</body>
```

Using console.log()


```
<body>

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

JavaScript Print


```
<body>

<button onclick="window.print()">Print this page</button>

</body>
```

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.
- Example
  - document.getElementById("demo").innerHTML = "Hello Dolly.";
  - (This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":)

Example

- <script>
  - let x, y, z; // Statement 1
  - x = 5; // Statement 2
  - y = 6; // Statement 3
  - z = x + y; // Statement 4
- document.getElementById("demo").innerHTML =
- "The value of z is " + z + ".";
- </script>

Semicolons ;

- Semicolons separate JavaScript statements.
- Add a semicolon at the end of each executable statement:
  - let 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

Examples

- 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 Keywords

var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

JavaScript Syntax

- JavaScript syntax is the set of rules, how JavaScript programs are constructed:
- // How to create variables:
  var x;
  let y;

- // How to use variables:
  x = 5;
  y = 6;
  let z = x + y;

JavaScript Values

- The JavaScript syntax defines two types of values:
  - Fixed values
    - Fixed values are called **Literals**.
  - Variable values
    - Variable values are called **Variables**.

JavaScript Literals

- The two most important syntax rules for fixed values are:
  - 1. **Numbers** are written with or without decimals:
  - 2. **Strings** are text, written within double or single quotes:

JavaScript Variables

- In a programming language, **variables** are used to **store** data values.
- JavaScript uses the keywords `var`, `let` and `const` 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:

4 Ways to Declare a JavaScript Variable:

- Using `var`
- Using `let`
- Using `const`
- Using nothing

- Note

- Variables are containers for storing values.

Example

- x = 5;
  y = 6;
  z = x + y;

- From all the examples above, you can guess:
  - x stores the value 5
  - y stores the value 6
  - z stores the value 11

JavaScript Identifiers

- ll JavaScript **variables** must be **identified** with **unique names**.
- These unique names are called **identifiers**.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).
- The general rules for constructing names for variables (unique identifiers) are:
- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

JavaScript Const

The `const` keyword was introduced in ES6 (2015).

Variables defined with `const` cannot be Redeclared.

Variables defined with `const` cannot be Reassigned.

Variables defined with `const` have Block Scope.

JavaScript Let

The `let` keyword was introduced in ES6 (2015).

Variables defined with `let` cannot be Redeclared.

Variables defined with `let` must be Declared before use.

Variables defined with `let` have Block Scope.

- Example
  - let x = "John Doe";

JavaScript Operators

- JavaScript Arithmetic Operators
- JavaScript Assignment Operators
- JavaScript String Operators
- JavaScript Comparison Operators
- JavaScript Logical Operators
- JavaScript Type Operators

JavaScript Arithmetic Operators

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

JavaScript Assignment Operators

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
>>=	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:

```
let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;
```

The result of text3 will be:

```
John Doe
```

JavaScript Comparison Operators

Operator	Description	Comparing	Returns	
==	equal to	x == 8	false	
		x == 5	true	
		x == "5"	true	
===	equal value and equal type	x === 5	true	
		x === "5"	false	
!=	not equal	x != 8	true	
!==	not equal value or not equal type	x !== 5	false	
		x !== "5"	true	
		x !== 8	true	
>	greater than	x > 8	false	
<	less than	x < 8	true	
>=	greater than or equal to	x >= 8	false	
<=	less than or equal to	x <= 8	true	

Logical Operators

Operator	Description	Example	
&&	and	(x < 10 && y > 1) is true	
||	or	(x == 5 || y == 5) is false	
!	not	!(x == y) is true	

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).
- Example :
  - function myFunction(p1, p2) {
    return p1 * p2; // The function returns the product of p1 and p2
    }

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.

Function Invocation

- 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)

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:

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

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

The result in x will be:

Functions Used as Variable Values

- Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.
  - 1. Local Variables
  - 2. Global Variables

JavaScript Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JavaScript Event Handlers

- Event handlers can be used to handle and verify user input, user actions, and browser actions:
- Things that should be done every time a page loads
- Things that should be done when the page is closed
- Action that should be performed when a user clicks a button
- Content that should be verified when a user inputs data
- And more ...
- Many different methods can be used to let JavaScript work with events:
- HTML event attributes can execute JavaScript code directly
- HTML event attributes can call JavaScript functions
- You can assign your own event handler functions to HTML elements
- You can prevent events from being sent or being handled
- And more ...

JavaScript Strings

- JavaScript strings are for storing and manipulating text.
- A JavaScript string is zero or more characters written inside quotes.
- Example :
  - let text = "John Doe";
  - You can use single or double quotes:

String Length

- Example
  - let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    let length = text.length;

Six other escape sequences are valid in JavaScript:

Code	Result
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tabulator
\v	Vertical Tabulator

JavaScript Arrays

- An array is a special variable, which can hold more than one value:
- Example :
  - const cars = ["Saab", "Volvo", "BMW"];

.

Why Use Arrays?

1. 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?

2. The solution is an array!
3. An array can hold many values under a single name, and you can access the values by referring to an index number
4. Creating an Array
5. Using an array literal is the easiest way to create a JavaScript Array

Syntax:

```
const array_name = [item1, item2, ...];
```

JavaScript Array Methods

- Popping and Pushing
- When you work with arrays, it is easy to remove elements and add new elements.
- This is what popping and pushing is:
- Popping items **out** of an array, or pushing items **into** an array.

JavaScript Array shift() , unshift(), length

The `shift()` method removes the first array element and "shifts" all other elements to a lower index.

The `unshift()` method adds a new element to an array (at the beginning), and "unshifts" older elements:

The `length` property provides an easy way to append a new element to an array

Splicing and Slicing Arrays

The `slice()` method creates a new array.

The `slice()` method does not remove any elements from the source array.

JavaScript Booleans

- A JavaScript Boolean represents one of two values: **true** or **false**.
- Boolean Values
- Very often, in programming, you will need a data type that can only have one of two values, like
  - YES / NO
  - ON / OFF
  - TRUE / FALSE
- For this, JavaScript has a **Boolean** data type. It can only take the values **true** or **false**.

JavaScript if, else, and else if

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

- Syntax :
- if (*condition*) {
  // block of code to be executed if the condition is true
  }

The else Statement

- 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
  }

The else if Statement

- 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
  }

JavaScript Switch Statement

- Syntax
- switch(*expression*) {
  case *x*:
  // code block
  break;
  case *y*:
  // code block
  break;
  default:
  // code block
  }

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- If there is no match, the default code block is executed.

The break Keyword

When JavaScript reaches a `break` keyword, it breaks out of the switch block.

This will stop the execution inside the switch block.
It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.

Note: If you omit the break statement, the next case will be executed even if the evaluation does not match the case.

JavaScript For Loop

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 statement creates a loop with 3 optional expressions:

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

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

Expression 2 defines the condition for executing the code block.

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

JavaScript For In

The JavaScript `for in` statement loops through the properties of an Object:

Syntax :

```
for (key in object) {
// code block to be executed
}
```

- Example Explained
- The **for in** loop iterates over a **person** object
- Each iteration returns a **key** (x)
- The key is used to access the **value** of the key
- The value of the key is **person[x]**

JavaScript For Of

The JavaScript `for of` statement loops through the values of an iterable object. It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more:

Syntax:

```
for (variable of iterable) {
// code block to be executed
}
```

variable - For every iteration the value of the next property is assigned to the variable. *Variable* can be declared with `const`, `let`, or `var`.

iterable - An object that has iterable properties.

JavaScript While Loop

- Syntax
- while (*condition*) {
  // code block to be executed
  }

The Do While Loop

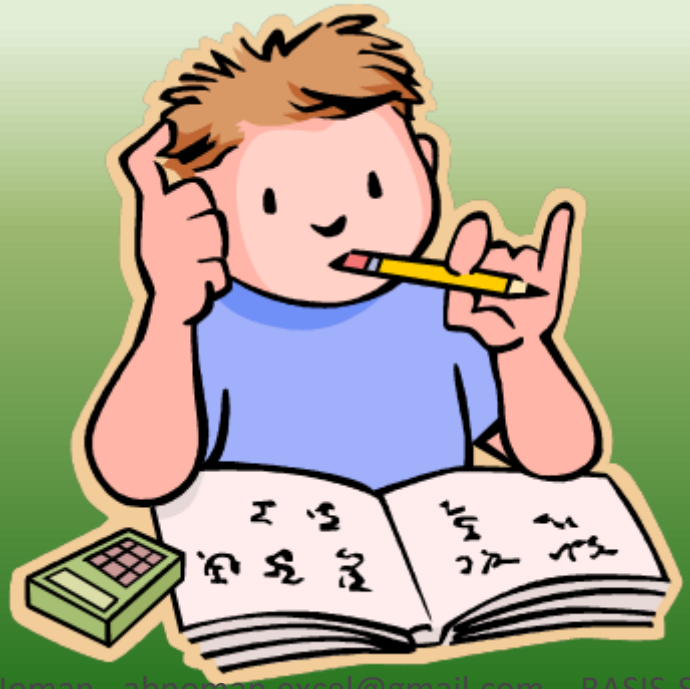
- Syntax
- do {
  // code block to be executed
  }
  while (*condition*);

JS (Javascript)

IS END??

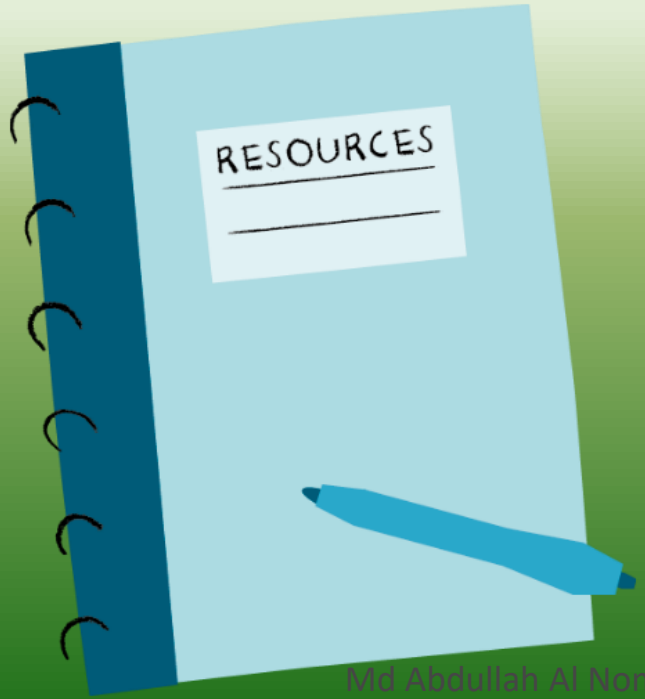
NEVER: JUST GO ON -> HTTPS://WWW.W3SCHOOLS.COM/JS/DEFAULT.ASP

Practice! Practice! Practice!!!



Md Abdullah Al Noman - abnoman.excel@gmail.com – BASIS-SEIP
Shabiha Ahmed - shabihaahmed96@gmail.com - BASIS SEIP

Resource



https://w3schools.com

https://google.com

Md Abdullah Al Noman - abnoman.excel@gmail.com – BASIS-SEIP
Shabiha Ahmed - shabihaahmed96@gmail.com - BASIS SEIP

thank you!

Md Abdullah Al Noman - abnoman.excel@gmail.com – BASIS-SEIP
Shabiha Ahmed - shabihaahmed96@gmail.com - BASIS SEIP