Jump2Learn - Publication™

Published By:

Ms. BHUMIKA K. CHARNANAND
Jump2Learn – Publication
Address:
112, Scarlet Shopping Mall, Opp. Prime Arcade, Adajan, Surat (India) - 395009.
Contact: +91 90 9999 09 60
Website: www.jump2learn.com
Email: info@jump2learn.com
Printed by: Moheet Printers, Surat.

Book Title: Advanced Web Technology
First Edition, 2021
ISBN: 978-93-92672-12-5
Price: ₹ 170/-

According to B.C.A., M.C.A., B.Sc. (IT, CS), M.Sc. (IT, ICT, CS), B.E. (IT), PGDCA, Std. 11(CBSE Board) and Diploma Courses of Engineering Programs of Various Universities

ADVANCED WEB TECHNOLOGIES

Authors:
Dr. Jagin M. Patel [M.C.A., M.Phil., Ph.D.]
M.K. INSTITUTE OF COMPUTER STUDIES, BHARUCH

Ms. Meghna N. Vithlani [M.C.A., SET]
M.K. INSTITUTE OF COMPUTER STUDIES, BHARUCH

[First Edition: 2021]

PREFACE

This book presents the concept of different Advanced Web Technologies like XML, JSON, JQuery, AJAX, NODE JS and all basic terminology related to the such web client.

This book provides numerous illustrations and examples with coding and output. Wherever necessary, pictorial descriptions of concepts are also included to facilitate the better understanding. This book is useful for acquiring the fundamental knowledge of all different web client technologies which are necessary for the development of any web based project.

As a reader of this book, you are most important critic and commentator. We value your opinion and want to know about the areas you'd like to see us publish in, what we could do better and any other words of wisdom you may be willing to pass our way.

Dr. Jagin Patel

Ms. Meghna Vithlani

ACKNOWLADGEMENTS

We express our sincere gratitude towards our family and friends for their unconditional support who are our greatest source of inspiration.

We would like to thank our management specially, who have provided all the relevant infrastructure to and encouraged us enduring this work.

We would specifically like to mention our heartfelt thanks to our colleagues whose direct or indirect support have helped us to complete this task.

The editorial team of Jump2Learn earns our genuine thanks for their support and guidance. Their continuous follow up and feedback made it possible for us to complete this task on time.

Dr. Jagin Patel

Ms. Meghna Vithlani

About Authors

Dr. Jagin M. Patel has 18+ years of experience in teaching field. Currently he is working as an Incharge principal at M.K. Institute Of Computer Studies, Bharuch. He holds Ph.D. and M.Phil. in Computer Science from Veer Narmad South Gujarat University, Surat. He holds MCA from the same University. He holds B.Sc. in Computer Science. He has 5 research papers and one book to his credit. His fields of interest are Image Processing, Machine Learning and Optical Character Recognition.

Ms. Meghna N. Vithlani has 18+ years of experience in teaching field. Currently she is working with M.K. Institute Of Computer Studies, a BCA College, Bharuch as an Asst. Prof. She is currently pursuing Ph.D. from Bhagwan Mahavir University, Surat. She is Gujarat State Eligibility Test (SET) qualified. She holds MCA degree from Veer Narmad South Gujarat University, Surat. She holds B.Sc degree from the same University . She has been well-liked among students for her knowledge based interactive teaching skills with an excellent communication expertise. She keeps her students engage by bringing them learn with real world. She has the ability to work in a constantly evolving environment and adjust her teaching methods based on the age of her students. Her core subjects are Programming Skills, Introduction To Computers, Web Designing, Software Engineering, RDBMS, PHP, Operating System, and Computer Graphics.

Index

Unit - 1

XML

1.1 Introduction

- XML is the acronym for eXtensible Markup Language.
- XML is simple text-based format. It is used to represent structured information such as documents, catalogue, books, student records etc.
- XML is a mark-up language based on tags within angled brackets, similar to HTML. Here tags are not predefined. We need to specify our own tags. These tags are self-explanatory.
- XML is designed to carry data, not to display it.

1.1.1 Characteristics of XML

There are many significant characteristics of XML that are beneficial for a range of systems.

i. Extensible: using XML, it is possible to create our own self-descriptive tags according application need.

ii. carries the data, does not present it : XML allows you to store the data regardless of how it will be displayed.

iii. Open standard: XML was created by the World Wide Web Consortium (W3C) and is freely available as an open standard.

iv. Structured format:

It is possible to precisely determine the arrangement, organisation and expressing of data in the file.

v. Validation: It is possible to validate XML file using document type declaration(DTD)

vi. Described format: In XML file, every piece of data has a name that is readable by both humans and machines.

vii. discoverable format: it is possible to parse XML file and understand the structure and relationships between the items

viii. platform independent: XML is a software and hardware-independent to send, receive or store data.

ix. XML is a universal format.

1.2 XML syntax (Declaration, Tags, elements)

1.2.1 XML declaration:

XML declaration sets parameters for basic XML parsing. It contains various parameters. The syntax of XML declaration is :

```
<?xml
        version = "version_information"
        encoding = "encoding_declaration"
        standalone = "standalone_status"
```

?>Here,

Parameter	Description	Value
Version	Specifies the XML version	1.0
Encoding	It specify encoding method used in the document. The default encoding is UTF-8.	UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9
Standalone	It tells the XML parser whether the document's content is based on data from an external source. The value is set to "no" by default. The value "yes" indicates no external declarations are necessary to parse the page.	Yes, no

Declaration examples:

Example 1: Following line declare XML document with version 1.0.

```
<?xml version = "1.0"  ?>
```

Example 2: Following line declare XML document with version 1.0, encoding UTF-8 and standalone = no.

```
<?xml
   version = "1.0"
    encoding = "UTF-8"
   standalone = "no" ?>
```

Example 3: Following example show XML declaration with all parameters defined in single quotes<?xml version = '1.0' encoding = 'UTF-8' standalone = 'no' ?>

Rules for XML Declaration/ XML document must meet these criteria:

i. XML document must begin with the XML declaration. XML declaration start with *<?xml* and end with *?>*, where "xml" is written in lower-case. XML declaration also contains various parameters.

ii. The sequence in which the parameters are placed is crucial. Version, encoding, and standalone are the correct order.

iii. The XML declaration ,if included, must be the first statement in the XML document.

iv. it must contain version number attribute

1.2.2 XML Tags

XML tags are the building blocks of an XML document. In XML, XML tags specify the scope of an element. They can also be used to add comments and specify environment-specific parameters.

XML Tag and HTML tags are similar, but XML is more flexible then HTML. HTML tags are predefined while XML tags are user defined.

(i) Starting Tag

Every XML element is marked by a start-tag. It is specified using tag name within angular brackets (<>) . For example:

<book>

(ii) Ending Tag

Every element has end-tag. It is specified using "/" before the name of an element in angular brackets. For example:

</book>

(iii) Empty Tag

An element which has no content is called as empty. It is denoted by in two ways:

(1) Nothing between staring tag and closing tag. In other words, start-tag immediately followed by an end-tag: For example

<book></book>

(b) tag name end with "/" within angular brackets. For example:

<book />

(iv) root tag:

There is exactly one root element in any XML document.All of the other elements are enclosed within it. The root element ir HTML is the < html> element, while root element can be anything in an XML file.

XML Tags Rules

i. A document can have exactly one root element.

ii. All attributes value must be put in single or double quotes

iii. Tag Names start with letters or underscore. Tag Names cannot start with a number or punctuation character. However, Tag Names can contain letters, numbers, and other characters.

iv. Comments can't be placed inside tags.

v. Tag Names cannot contain spaces.

vi. Tags are case-sensitive. For example, <book>, <Book>, <BOOK> all are different.

vii. Tag names cannot start with word *xml* ,in lowercase, uppercase, or mixed.

1.2.3 XML elements:

- XML elements are the building blocks of an XML document. The XML Element is collection of opening-tag, data and closing-tag. i.e. it contain everything from the element's start tag to the element's end tag.
- Each XML document contains one or more elements.
- Elements can't overlap. In other words, element should properly nested.
- All xml elements must have a closing tag.

Elements contain following:

- text
- attributes
- other elements
- mix of the above

Example of XML elements:

Following is the examples of XML element:

<title>Advanced in web technology </title>

<publisher isdn="101175" > jump2learn </publisher>

<street_name> bari faliya </street_name>

1.3 XML document

XML documents contain only markup and content. An XML document can contains wide variety of data.

An XML document consists of two sections: prolog and body:

(i) prolog:

It includes an XML declaration and an optional reference to external structuring documents

Prolog Section is first and most important part of an XML Document. It indicates that document is marked up in XML.

XML Prolog includes:

- XML declaration,
- DOCTYPE and comments,
- information about the character encoding used, processing instructions,
- reference to either a document type definition (DTD) or XML Schema

The prolog is an optional section of the XML document. If a prolog section is included in an XML document, it must appear before the root element.

This section contains two parts:

i. XML declaration. (it was discussed in previous section)

ii. Document Type Definition (DTD): provide advanced settings

These both parts are optional.

Example of prolog:

Following example illustrates the prolog:

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE booklist SYSTEM "booklist.dtd">

(ii)body:

Body consist of a number of elements.

Figure 1.1 shows example of XML document and its various section.

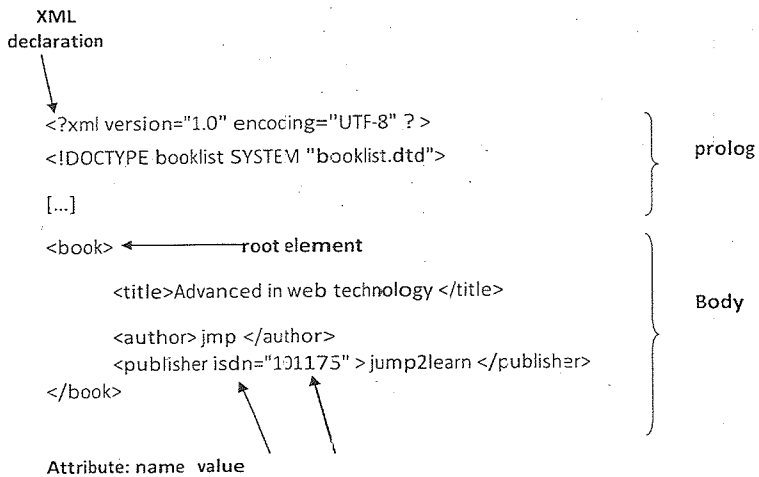


Figure 1.1: example of XML document

Document Type Definition (DTD)

Document Type definition is used to specify the set of rules or instructions for current XML Document structure as per the requirement. Set of rules or instructions can be defined in a separate file and this file can be used to validate XML document.

DTD define entities or default attributes values. It supports validation, a special mode of parsing that checks grammar and vocabulary of mark-up.

1.4 XML Comments

XML comments are same as HTML comments. Comments are written with in <!-- and -->

Following is comment

```
<!-- this is comment -->
```

Example of XML document

Example 1:

```
<?xml version="1.0" ?>
    <Color>
        <!-- this is comment -->
        <Option1 Name='Blue'/>
        <Option Name="Yellow"/>
        <Option Name="Crimson"/>
    </Color>
```

Example 2:

```
<?xml version="1.0" ?>
<address>
    <street_no> 10 </street_no>
    <street_name> bari faliya </street_name>
    <district> bharuch </district>
</address>
```

Example 3: nested element

```
<?xml version='1.0' encoding='UTF-8'?>
<producers>
<producer type='assistant'>
    <name>MANYA</name>
    <age>20</age>
    <contact info='mobile'>1111</contact>
</producer>
<producer type='manager'>
    <name>NAVYA</name>
    <age>25</age>
    <contact info='office'>252525</contact>
</producer>
<producer type='junior'>
    <name>SHUBH</name>
    <age>32</age>
    <contact info="resi">8787</contact>
</producer>
</producers>
```


Example 4: In following example is <catalogue> is root element. All elements in the document are contained within <catalogue>. The <book> element has 2 children: <title> and <price>.

```
<?xml version="1.0" ?>
<catalogue>
<book>
<title> Fundamentals of web technology </title>
<price> 150 </price>
</book>
<book>
<title> Advanced of web technology </title>
<price> 200 </price>
</book>
<book>
<title> ASP.net </title>
<price> 200 </price>
</book>
</catalogue>
```

Questions:

1. What is XML?
2. Write Characteristic XML.
3. Write note on XML document prolog section.
4. Write note on XML document element section
5. Write rules of XML declaration with example

MCQ

Select correct answer.

1.XML stand for:

A. eXtra Markup Language

B. eXtensible Markup Language

C. Example Markup Language

D. X-Markup Language

Ans: B

2. Which statement is true about XML?

A. All XML elements must have a closing tag

B. All XML elements must be lower case

C. All XML documents must have a DTD

D. All given

Ans: A

3. correct syntax of defining the XML version is:

A. <xml version="version_no" />

B. <?xml version=" version_no " ?>

C. <?xml version=" version_no " />

D. None of the above

Ans: B

4. How to give Comment in XML document?

A. <?-- -->

B. <!-- --!>

C. <!-- -->

D. </-- -- >

Ans: C

5. DTD stand for:

A. Direct Type Definition

B. Document Type Definition

C. Document Type Declaration

D. Dynamic Type Definition

Ans: B

6. Which of the following XML is correct?

A. <myElement myAttribute="some_Value"/>

B. <myElement myAttribute='some_Value'/>

C. <myElement myAttribute='some_Value">

D. <myElement myAttribute="some_Value'/>

Ans: A

7. Can attributes of XML element have multiple values?

A. Yes B. No

Ans: B

8. Which of the following XML fragments are well-formed?

A. <myElement myAttribute="some_Value"/>

B. <myElement myAttribute='some_Value'/>

C. <myElement myAttribute='some_Value">

D. <myElement myAttribute="some_Value'/>

Ans: A

9. How to add the attribute named Title to the <book> tag?

A. <book attribute Title ="advanced web">

B. <book Title attribute =" advanced web">

C. <book Title attribute_type=" advanced web">

D. <book Title =" advanced web" >

Ans: D

10. XML tags are Case Sensitive:

A. true B. false

Ans: A true

11. it easier to process XML than HTML

A. true B. false

Ans: A true

12. XML document is Well formed if _____

A. contain an comment

B. it contains a root element

C. it contains one or more elements

D. must contain one or more elements and root element must contain all other elements

Ans D

13. Which statement is true about XML?

A. All XML elements must be properly closed

B. All XML elements must be lower case

C. All XML documents must have a DTD

D. All given

Ans: A

14. Which statement is NOT true in reference to XML?

A. White-space is not preserved in XML

B XML elements can be empty

C. XML elements must be properly nested

D. XML documents must have a root tag

Ans: A

15. XML preserves white spaces: true or false?

A. true B. false

Ans: A. true

15. XML elements cannot be empty

A. true B. false

Ans: B

17. XML attribute values must always be enclosed in quotes

A. true B. false

Ans: A. true

13. What is the correct syntax of the declaration which defines the XML version?

A. <?xml version="1.0"?>

B. <xml version="1.0" />

C. <?xml version="1.0" />

D. <xml version="1.0" ?>

Ans: A. <?xml version="1.0"?>

19. Can you say following is a "well formed" XML document?

<?xml version="1.0"?>

<book>

<title>advanced web</ title >

<author>JMP</author >

</book>

20. A. yes B. no

Ans: A. yes

21. Which is not a correct name for an XML element?

A. <book name>

B. <Age>

C. <NAME>

D. None

Ans: A. <book name>

Unit - 2

jQuery Fundamentals

2.1 Introduction and Basics

- jQuery is a JavaScript library. It helps to create Web Pages and applications quickly and easily.
- It is a lightweight JavaScript and AJAX library, which support multiple browsers and emphasizes on the interaction between JavaScript and HTML.
- It enables to create the Web Applications that contain animations, communicate with server to send requests or get response and handle events.
- jQuery is available as .js file as it is written in JavaScript language.
- It supports and works on many different browsers like IE, Mozilla, Google Chrome, and Safari.
- So, it is not required to write different JavaScript code for each browser individually.
- It allows to select DOM elements, traverse or navigate them and modify their content.
- It allows to capture a wide variety of events, such as clicking a button, pressing a key from keyboard, moving a mouse etc.
- It provides built-in animation effects that can be included in websites to make them more attractive and dynamic.
- It can also be integrated with AJAX to create a more spontaneous website in sending and receiving the request and responses from the server.

2.1.1 Advantages

Some advantages of using jQuery are:

- Animation effects like fading, sliding, hiding and others to HTML elements can be added.
- It enables to make AJAX requests to web server to add data without reloading the page.
- It allows to manipulate DOM by adding, removing and reordering the content of Web page.

- It allows to create animated images slideshows, animated multi-level dropdown menus, and drag and drop interfaces.
- It allows to create complex forms with client-side validations and auto-complete AJAX text fields that retrieve data from server-side database.

2.1.2 Loading and Using jQuery

- jQuery is available as a single file with extension .js.
- The latest version of jQuery is 3.6. It is available in two formats compressed and uncompressed.
- Both the formats can be downloaded from jQuery.com.
- In this book we will be considering the version 3.5.1.js file for the explanation and examples.
- The following code shows how to include the jQuery library in a Web Page:

```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

The *src* attribute of the *<script>* specifies the path of the jQuery file.

- If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).
- Google is an example who host jQuery
- Following is the way to use Google to host jQuery:

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
</head>
```

2.1.3 jQuery Syntax

- With jQuery we select the HTML elements and perform some "actions" on them.

  Basic syntax is: $(selector).action()
  - A $ sign - define/access jQuery,
  - A (selector) - "query (or find)" HTML elements

- action() to be performed on the element(s)

- Example:

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
```

In the above example, all the jQuery methods are wrapped inside document ready event. This will prevent the jQuery methods to be executed before the document is finished loading.

2.1.4 jQuery Selectors

- jQuery selectors allow to select and manipulate HTML element(s).
- They can select or manipulate element as a group or as a single element.
- They find the HTML elements based on their ids, classes, types, attributes or it's values and much more.
- They also support CSS syntax.
- All selectors in jQuery start with the dollar sign and parentheses as $().
- Example:

  $("p").css("background-color:green");

List of selectors are as below:

Selectors	Example	Selects
*	$("*")	Returns all the elements
#id	$("#test")	Returns all elements with the specified id test
.class	$(".intro")	Returns all elements with class intro
element	$("p")	Returns all p elements
:first	$("p:first")	Returns the first p elements

:last	$("p:last")	Returns the last p elements
:even	$("tr:even")	Returns all tr elements at the even number
:odd	$("tr:odd")	Returns all tr elements at the odd number
:eq(index)	$("ul li:eq(3)")	Returns the fourth element in a list, here index starts at 0
:header	$("header")	Returns all header elements such as h1,h2,....h6
[attribute]	$("[href]")	Returns all elements having href attribute
[attribute=value]	$(["href=first.html"])	Returns all elements with href attribute having value equal to first.html.
:input	$(":input")	Returns all input elements
:text	$(":text")	Returns all input elements with the type, text
:password	$(":password")	Returns all input elements with the type, password
:checkbox	$(":checkbox")	Returns all input elements with the type, checkbox
:radio	$(":radio")	Returns all input elements with the type, radio
:submit	$(":submit")	Returns all input elements with the type, submit
:reset	$(":reset")	Returns all input elements with the type, reset
:button	$(":button")	Returns all input elements with the type, button
:image	$(":image")	Returns all input elements with the type, image
:file	$(":file")	Returns all input elements with the type, file
:enabled	$(":enabled")	Returns all enabled input elements
:disabled	$(":disabled")	Returns all disenabled input elements
:checked	$(":checked")	Returns all checked input elements
:selected	$(":selected")	Returns all selected input elements

Example:

```
$("document").ready(function(){
$("button").click(function(){
    $(".test").hide();
$("#p1").hide();
```

2.2 jQuery Effects

- jQuery enables us to add various effects on a web page which could be like loading images after certain event.
- jQuery has different effects like fading, sliding, show/hide, animate/stop, and others.

2.2.1 jQuery hide() , show() & toggle() Events

- The hide() and show() methods are used to hide and show the elements respectively:
- The toggle() method is used to toggle between the hide() and show() methods.

Syntax:

```
$(selector).hide(speed,callback);
$(selector).show(speed,callback);
$(selector).toggle(speed,callback);
```

- The speed parameter specifies the speed of the hiding/showing, it can take the values as either "slow", "fast", or milliseconds. It is optional.
- The callback parameter is a function to be executed after the completion of hide() or show() or toggle() methods. It is optional.

Example:

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("#hide").click(function(){
    $("p").hide();
  });
  $("#show").click(function(){
    $("p").show();
```



```
  });
  $("#toggle").click(function(){
      $("p").toggle();
  });
});
</script>
</head>
<body>
<p>
<b>Advanced Web Technologies <br>
Published by Jump2Learn</b>
</p>
<button id="hide">Hide</button>
<button id="show">Show</button>
<button id="toggle">Toggle</button>
</body>
</html>
```

2.2.2 jQuery Fading Methods

- The fading methods of an element enables it's in and out of visibility.
- jQuery has the following fade methods:
  - fadeIn()
  - fadeOut()
  - fadeToggle()
  - fadeTo()

Syntax:

```
$(selector).fadeIn(speed,callback);
$(selector).fadeOut(speed,callback);
$(selector).fadeToggle(speed,callback);
$(selector).fadeTo(speed,opacity,callback);
```

- The speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds. It is optional.
- The opacity parameter in the fadeTo() method specifies fading to a given opacity (value between 0 and 1). It is optional.
- The callback parameter is a function to be executed after the function completes. It is optional.
- **Example:**

```
<html>
<head>
<script src="jquery-3.5.1.js"></script>
<script>
$(document).ready(function(){
    $("#bt1").click(function(){
        $("#div1").fadeIn();
        $("#div2").fadeIn(1000);
    });
    $("#bt2").click(function(){
        $("#div1").fadeOut();
        $("#div2").fadeOut(1000);
    });
    $("#bt3").click(function(){
        $("#div1").fadeToggle();
        $("#div2").fadeToggle(1000);
    });
    $("#bt4").click(function(){
        $("#div1").fadeTo("slow",0.3);
        $("#div2").fadeTo(1000,0.15);
    });
});
</script>
```



```
<style type="text/css">
#div1{width:80px;
    height:80px;
    background-color:green;
    display:none;
}
#div2{width:80px;
    height:80px;
    background-color:red;
display:none;
}
</style>
</head>
<body>
<p> Fading effect demonstration on jump2learn</p><br>
<button id="bt1">FadeIn</button>
<button id="bt2">FadeOut</button>
<button id="bt3">FadeToggle</button>
<button id="bt4">FadeTo</button><br><br><br>
<div id="div1"></div><br><br>
<div id="div2"></div>
</body>
</html>
```

2.2.3 jQuery Sliding Methods

- With jQuery you can create a sliding effect on elements.
- jQuery has the following slide methods:
    - slideDown()
    - slideUp()
    - slideToggle(

Syntax:

$(*selector*).slideDown(*speed,callback*);

$(*selector*).slideUp(*speed,callback*);

$(selector).slideToggle(speed,callback);

- The speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds. It is optional.
- The callback parameter is a function to be executed after the sliding completes. It is optional.
- **Example:**

```
<html>
<head>
<script src="jquery-3.5.1.js"></script>
<script>
$("document").ready(function(){
    $("#flip").click(function(){
        $("#panel").slideToggle(1000);
    });


});
</script>
<style>

#flip { background-color:grey;
color:white;
text-align:center;
    border:solid 5px yellow;
    padding:5px}
#panel{background-color:grey;
color:white;
```



```
    text-align:left;
    border:solid 5px yellow;
    padding:100 px;
    display:none;}
</style>
</head>
<body>
<h1 align="center"> Demonstrate Sl ding Effect</h1>
<div id="flip"> Click to Slide the Panel Down</div>
<div id="panel"> Faculty <br>
                Student<br>
                Administrator<br>
</div>
</body>
</html>
```

2.2.4 jQuery Animations

- The jQuery animate() method is used tc create custom animations.
- **Syntax:**

  $(selector).animate({params},speed,callback);
- The params parameter defines the CSS properties to be animated. It is required.
- The speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds. It is optional.
- The callback parameter is a function to be executed after the animation completes. It is optional.
- jQuery stop() Method
- The jQuery stop() method is used to stop an animation or effect before it is finished.
- The stop() method works for all jQuery effect functions, including sliding, fading and custom animations.

Syntax:

$(selector).stop(stopAll,goToEnd);

- The stopAll parameter specifies whether the animation queue should be cleared or not. Default is false, which means that only the active animation will be stopped, allowing any queued animations to be performed afterwards. It is optional.
- The goToEnd parameter specifies whether or not to complete the current animation immediately. Default is false. It is optional.
- So, by default, the stop() method kills the current animation being performed on the selected element.
- **Example:** The following example demonstrates the animate() and stop() methods, with no parameters:

```
<html>
<head>.
<script src="jquery-3.5.1.js"></script>
<script>
$("document").ready(function(){
    $("#start").click(function(){
        $("#div1").animate({left:'500px',height:'300px',width:'300px',opacity:
        '0.4'},2000);
    });
    $("#stop").click(function(){
        $("#div1").stop();
    });
});
</script>
<style type="text/css">
#div1{
    background:green;
    height:100px;
    width:100px;
    position:absolute;
}
```



```
</style>
</head>
<body>

<button id="start">Start Animation</button>
<button id="stop">Stop Animation</button>

<p>This is advanced web technolcgy books </p>

<div id= "div1" style=""></div>
</body>
</html>
```

2.2.5 jQuery Callback Functions

- JavaScript statements are executed one line at a time in sequence.
- But when we use jQuery effects, the next line of code can be run even though the effect is not finished. This can create errors.
- To prevent such thing a callback function is used.
- A callback function is executed after the current effect is finished completely.

Syntax:

```
$(selector).hide(speed,callback);
```

Example:

- The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

```
$("button").click(function(){
$("p").hide("slow",function(){
alert("The paragraph is now hidden");
                    })
                });
```

- The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed:

```
$("button").click(function(){
$("p").hide(1000);
  alert("The paragraph is now hidden");
});
```

2.2.6 jQuery - Chaining

- Chaining allows to run multiple jQuery methods (on the same element) within a single statement.
- It also allows to run multiple jQuery commands, one after the other, on the same element(s).
- To give chaining effect, simply append the action to the previous action.
- **Example:** The following example chains together the css(), slideUp(), and slideDown() methods

```
$("#p1").css("color","red")
.slideUp(1000)
.slideDown(1000);
```

2.3 jQuery Manipulation methods

- jQuery provides various methods like html(), append, get() and many more to modify or manipulate the content of HTML element and attributes.
- Next, we will discuss different jQuery methods for the manipulations of content or attribute values in html.

2.3.1 jQuery Get/Set methods

- Following methods of jQuery are used for DOM manipulation:

Method	Description
text()	sets or gets text of selected elements
val()	sets or gets value of a form field

html()	sets or gets the content of selected elements (including HTML markup)
attr()	sets or gets attribute values.

- **Example:**Following example demonstrates the use of jQuery methods text(), val(), html() and attr() for retrieving purpose.
- Here on clicking different buttons :
  - text() method will retrieve the content of paragraph having id p1 excluding any html tags within it.
  - html() method will retrieve the content of p1 including the tags <b> , <i>
  - val() will retrieve the value of textbox having id t1 and
  - attr() will retrieve the value of attribute href of anchor tag <a>having id a1

```
<html>
<head>
<script src="jquery-3.5.1.js"></script>
<script>
  $("document").ready(function(){
      $("#btn1").click(function(){
          alert("TEXT:" + $("#p1").text());
      });
      $("#btn2").click(function(){
          alert("HTML:" + $("#p1").html());

      });
      $("#btn3").click(function(){
          alert("Value of textbox is:" + $("#t1").val());

      });
      $("#btn4").click(function(){
          alert($("#a1").attr("href"));
      });
```

```
    });
</script>
</head>
<body>

<p id="p1"> This is some <b> bold </b> text in the<i>paragraph</i></p><br>
Enter your city:<input type="text" value="bharuch" id="t1"><br><br><br>

<a href="jump2learn_homepage.html" id="a1">Click here</a>
<input type="button" id="btn1" value="Show text">
<input type="button" id="btn2" value="Show html">
<input type="button" id="btn3" value="Show value">
<input type="button" id="btn4" value="Show href">

</body>
</html>
```

Example: Following example demonstrates the use of jQuery methods text(), val(), html() and attr() for setting different values.

- Here on clicking different buttons :
  - text() method will set the new content of paragraph having id p1 excluding any html tags within it.
  - html() method will set the new content of p1 including the tags <b>
  - val() will set the new value of textbox having id t1 and
  - attr() will set the new value of attribute href and title of anchor tag <a>


```
<html>
<head>
<script src="jquery-3.5.1.js"></script>
<script>
   $("document").ready(function(){
      $("#btn1").click(function(){
         $("#p1").text("Jump2Learn!");
      });
      $("#btn2").click(function(){
         $("#pt2").html("<b>Jump2Learn!</b>");
      });
      $("#btn3").click(function(){
         $("#t1").val("BHARUCH");
      });
         $("#btn4").click(function(){
           $("a").attr({"href":"anotherpage.html","title":"set another
           page"});
         });
   });
</script>
</head>
<body>
<p id="p1">This is aadvance web technology book.</p>
<p id="p2">This is secondparagraph.</p><br>
<a href="first.html" title="first_page">Click to check</a>
<p>Input field: <input type="text" id="t1" value="SURAT"></p>
<button id="btn1">Set Text</button>
<button id="btn2">Set HTML</button>
<button id="btn3">Set Value</button>
<button id="btn4">Set HREF Attribute</button>
</body></html>
```

2.3.2 jQuery Insert methods

- Following methods of jQuery are used to add new content

Method	Description
append()	Inserts content at the end of the selected elements
prepend()	Inserts content at the beginning of the selected elements
after()	Inserts content after the selected elements
before()	Inserts content before the selected elements
wrap()	wraps specified HTML element(s) around each selected element

- **Example:** Following example demonstrates the use of jQuery insert methods.

```
<html>
<head>
<script src="jquery-3.5.1.js"></script>
<script>
$("document").ready(function(){
  $("#btn1").click(function(){
      $("p").append(" <b>Best book</b>.");
  });
  $("#btn2").click(function(){
      $("ol").append("<li>Mobile Development</li>");
  });
  $("#btn3").click(function(){
      $("p").prepend(" <b>Best book</b>.");
  });
  $("#btn4").click(function(){
      $("ol").prepend("<li>Mobile Development</li>");
  });
  $("#btn5").click(function(){
```



```
      $("img").before("<b>Text Appears Before Logo</b>");
  });
  $("#btn6").click(function(){
      $("img").after("<i>Text Appears After Logo</i>");
  });
  $("#btn7").click(function(){
      $("p").wrap("<div></div>");
  });

});
</script>
<style type="text/css">
div{background-color: orange; border:black 2pt solid}
</style>
</head>
<body>
<p>This is an advanced web technology book.</p>
<p>It is published by Jump2Learn publisher.</p>
<ol>
<li>Web Design1</li>
<li>Java</li>
<li>.Net</li>
</ol>
<imgsrc="jump2learn logo.jpg" width="100"height="140"><br>

<button id="btn1">Append text</button>
<button id="btn2">Appended list items</button>
<button id="btn3">Prepend text</button>
<button id="btn4">Prepended list items</button>
<button id="btn5">Insert before</button>
<button id="btn6">Insert after</button>
```

```
<button id="btn7">Wrap Element </button>
</body>
</html>
```

2.3.3 jQuery Removeelements:

- Following methods of jQuery are used to remove element and content :

Method	Description
empty()	Removes the child elements from the selected element
remove()	removes the selected element(s) and its child elements.
unwrap()	removes the parent element of the selected elements

- **Example:** Following example demonstrates the use of jQuery remove methods

```
<html>
<head>
<script src="jquery-3.5.1.js"></script>
<script>
$("document").ready(function(){
   $("#btn1").click(function(){
   $("#div1").empty();
   });
   $("#btn2").click(function(){
   $("#div1").remove();
   });
   $("#btn3").click(function(){
   $("p").unwrap();
   });
});
</script>
</head>
```



```
<body>
<div id="div1"style="height:100px;width:300px;border:solid red 5px;background-
color:yellow;">
This is some text in the div.
<p>This is first paragraph in the div.</p>
<p>This is another paragraph in the div.</p>
</div>
<br>
<p><b>This paragraph is out of div</b></p>

<button id="btn1">empty element</button>
<button id="btn2">remove element</button>
<button id="btn3">unwrap div element</button></body>
</html>
```

2.3.4 jQuery css() Method

The css() method sets or returns one or more style properties for the selected elements.

Get a CSS Property

To get or retrieve the value of a specified CSS property, use the following syntax:

```
css("propertyname");
```

The following example will return the background-color value of the FIRST matched element:

```
<html>
<head>
<script src="jquery-2.1.4.js"></script>
<script>
$("document").ready(function(){
   $("#btn1").click(function()
   {
```

```
        alert($("h1").css("background-color"));
    });
});
</script>
</head>
<body>
<h1 style="background-color:green;">Advanced Web Technologies</h1>
<p>Welcome to jump2learn</p>
<button id="btn1"> Get CSS Property</button>
</body>
</html>
```

Set a CSS Property

To set a specified CSS property, use the following syntax:

```
css("propertyname","value");
```

The following example will set the background-color value for ALL matched elements:

```
<html>
<head>
<script src="jquery-3.5.1.js"></script>
<script>
$("document").ready(function(){
    $("#btn1").click(function()
    {
    $("h1,p").css("background-color", "yellow");
    });
});
</script>
</head>
```



```
<body>
<h1>Advanced Web Technologies</h1>
<p>Welcome to jump2learn</p>
<button id="btn1"> Set CSS Property</button>
</body>
</html>
```

Set Multiple CSS Properties

To set multiple CSS properties, use the following syntax:

```
css({"propertyname":"value","propertyname":"value",...});
```

The following example will set a background-color and a font-size for ALL matched elements:

```
$("h1, p").css({"background-color": "yellow", "font-size": "200%"});
```

2.4 Exercise

2.4.1 Choose the correct option(s) for the following MCQs

1. Which of the following jQuery method is used to stop jQuery for few milliseconds?
    A. stop() method
    B. delay() method
    C. slowdown() method
    D. pause() method

    Ans: A
2. Which jQuery method is used to set one or more style properties to the selected element?
    A. The html() method
    B. The style() method
    C. The css() method

D. All 3 options

Ans: C

3. Which of the following sign is used as a shortcut for jQuery?
   A. the % sign
   B. the & sign
   C. the $ sign
   D. the @ sign

   Ans: C

4. Which tag is used to import the jQuery library file in the code?
   A. <head>
   B. <script>
   C. <import>
   D. <style>

   Ans: B

5. Which selector expression will hide all the elements having class cute?
   A. $(.cute).hide();
   B. $(".cute").hide();
   C. $("#cute").hide();
   D. $(".cute").hides();

   Ans: B

6. Pick the odd one out.
   A. fadeIn()
   B. fadeOut()
   C. show()
   D. fadeTo()

   Ans: C

7. Which of the following jQuery method is used to sets or gets the value of form control?
   A. html()

   B. text()
   C. value()
   D. val()

   Ans: D

8. The _____ method removes the child elements from the selected element whereas the _____ method removes the selected element and its child elements.
   A. empty() and remove()
   B. remove() and empty()
   C. delete() and remove()
   D. remove() and delete()

   Ans: A

9. Which of the following is not a jQuery event?
   A. click()
   B. onClick()
   C. blur()
   D. mouseleave()

   Ans: B

10. Which of the following selector expression will select the element having id "abcd"?
    A. $(#abcd)
    B. $('#abcd')
    C. $('.abcd')
    D. $(abcd)

    Ans: B

11. Which of the following method does not provides sliding effect?
    A. slideUp()
    B. slideDown()
    C. fadeIn()
    D. slideToggle()

    Ans: C

12. I have a dropdown menu in my form having class property "city". What line of jQuery code should I write if I want to get the value of that menu? Consider that I want to store that value in a variable named "vcity".
    A. $vcity = $('.city').val();
    B. var vcity = $('.city').val();
    C. var vcity = $('.city').text();
    D. var vcity = $('.city').html();

Ans: B

2.4.2 Answer the following in brief:

1. What is the use of callback parameter of JQuery effect methods?
2. What is the d fference between remove() and empty() method of JQuery?
3. What do you mean by Document Ready Event?
4. What is element selector in JQuery?
5. Explain #id selector in JQuery
6. What is .class selector in JQuery?
7. What is GET method in JQuery?
8. What is SET method in jQuery?
9. What is chaining effect in jQuery?
10. Give difference between wrap and unwrap methods in jQuery.

2.4.3 Give the answer in detail:

1. Write a note on jQuery.
2. Write a note on jQuery events.
3. Write a note on effects of jQuery.
4. What are the types of selectors in jQuery? Explain each with
5. Explain jQuery methods for CSS manipulation.
6. Explain jQuery nsert methods.
7. Explain jQuery methods to modify DOM.

Unit - 3 JSON (JavaScript Object Notation)

3.1 What is JSON

- JSON stands for Java Script Object Notation. It is a lightweight text data interchange format. It is format for storing and exchanging data.
- The extension of JSON file is .json.
- The JSON is platform-independent.
- It is extended from the JavaScript scripting language. It uses JavaScript syntax to describe data objects.

3.1.1 Features of JSON

I. Self-Describing: It is "self-describing" and easy to understand
II. Parse: It is easy for machines to parse
III. Openness: It is open source and free to use.
IV. Simplicity: It is easy to read and write.
V. Scalable: It supports almost every kind of language, framework, and library.
VI. It is an easier-to-use than XML.
VII. It is a lightweight data-interchange format
VIII. It is language independent
IX. JSON is a text-based, human-readable data exchange format
X. It is used to transmit data between a server and web applications.

3.2 Comparison of JSON and XML

3.2.1 Similarities between the JSON and XML

The following are the Similarities between the JSON and XML:

(i). **Hierarchical structure:** Both support hierarchical structure (values within values)
(ii). **Self-descriptive:** Both are self-descriptive as they are human-readable text.
(iii). **Data interchange format:** Both can be used as data exchange by variety of programming languages.

(iv). **Retrieve data:** Both can be used to retrieve data from a web server. Both formats can be retrieved by using HTTP requests. Both can be fetched with an XMLHttpRequest.
(v). **Parse:** Both formats are simple to parse
(vi). **Open source:** Both are open source.

3.2.2 Differences between the JSON and XML.

The following are the differences between the json and xml:

Comparison Parameter	JSON	XML
Full form	Javascript Object Notation	Extensible Markup Language
extension	.json	.xml
media type	application/json.	application/xml or text/xml.
extended	It is extended from javascript.	It is extended from SGML.
Supported Data type	strings, numbers, Booleans, null, array.	does not have any type.
tags	JSON has no tags.	XML data is represented in tags
File size	File size is smaller	XML file is larger.
Quicker	it is quicker to read/write.	XML file takes time to read/write
Array support	Support arrays to represent the data.	Does not support the concept of arrays.
parser	It can be parsed using a standard javascript function.	Needs an XML parser respective to their programming language to use that.
Ease of parsing	It is easily parsed	It is difficult to parse.
secure	It is less secure than XML.	It is more secure than JSON.

Namespaces support	It does not provide any support for namespaces.	It supports namespaces.
Supported encoding	It supports only UTF-8 encoding.	It supports various encoding.
	It is data-oriented.	It is document-oriented.

3.3. JSON supported data types

Type	Description
Number	Double- precision floating-point number. Example: 12, 345,101175
String	It is sequence of characters in double-quote. Example: "Shubh", "shaurya"
Boolean	Value is true or false
Array	It is an ordered sequence of values
Object	It is an unordered collection of key:value pairs
null	empty

3.4 JSON objects

A JSON object made up of data in the form of key-value or attribute-value pair. It can have zero, one, or more key-value pairs, which are also known as properties. Key-value is separated by colon (:). The key is a string that uniquely identifies the key-value pair and the values are the JSON types. A comma separates each key-value pair. The key-value pair's order is unimportant. The object is surrounded by curly braces {}.

Syntax of data: key:value

Following examples represent data in JSON

Example 1: "Name":"swara"

Example 2: "price":200

Empty Object: Empty object has nothing between curly braces {}.

Object with value:

Following represent JSON object having title attribute with string value, and attribute price with numeric value.

{ "title" : "advanced concept", "price":200 };

Following code is Creation of an object book having title attribute with string value, and attribute price with numeric value.

var book= { "title" : "advanced concept", "price":200 };

It can also be written as:

var book= { "title" : "advanced concept",
"price":200 };

3.5 JSON Arrays

- Arrays in JSON are almost the same as arrays in JavaScript.
- A JSON array is an ordered list of values.
- It saves multiple values of a specific JSON data type.
- Values inside an array are separated by a comma.
- JSON arrays written in side [ ] (square bracket).
- The array index starts with 0.

3.5.1 JSON Array of Strings

JSON Array of strings is an ordered list of values with string type.

Syntax:

["string1","string2","string3",...]

Following are the examples of JSON arrays of string values:

["red", "green", "blue"]

["September", "October", "November", December"]

["manya","navya","yash","manushi","haranish","manav","swara"]

3.5.2 JSON Array of Numbers

JSON Array of Numbers is an ordered list of number values. Following are the examples of JSON arrays of Numbers.

```
[25,35, 45]
[30,40,80,44,89,85,26]
```

3.5.3 JSON Array of Booleans

JSON Array of Booleans is an ordered list of Booleans values. Following are the examples of JSON arrays of Booleans.

```
[false, true, false, true, false]
[true, false]
```

3.5.4 JSON Array of Objects

JSON Array of Objects is an ordered list of objects. Following is the example of JSON array having three Objects. In the example, the object "Book" is an array containing three objects. Each object is a record of a book (with a title and price).

```
{"Book":   [{ "title" : "Advanced Web Technologies", "price":200 },
            { "title" : "Fundamental of web", "price":140 },
            { "title" : "Advanced concept", "price":300 },
]}
```

The first title in the object array can be accessed like this:

```
Book[0].title   ( remember: The array index starts with 0)
        OR
Book [0]["title"]
```

How to Assign/ modify value:

To assign value use syntax: array_name[index]. Attribute=value.

Following example assign value "Advanced Web Technologies" to "title" of first object.

```
Book[0].title = "Advanced Web Technologies";
Book [0]["title"] = " Advanced Web Technologies";
```


3.5.5 JSON Multidimensional Array

It is possible to store an array inside another JSON array. This concept is known as an array of arrays or a multi-dimensional array.

Following is the example of multi-dimensional array:

```
[
    [ "a1", "a2", "a3" ],
    [ "b1", "b2", "b3" ],
    [ "c1", "c2", "c3" ]
]
```

Following example Create multidimensional array object

```
var books = { ".net" : [
            { "title"  : ".net complete", "price" : 200 },
            { " title "  : ".net black book", "price" : 300 }
    ],
    "web"  : [
     { " title " : "concepts of web technology ", "price": 150 },
     { " title " : "Advanced Web Technologies", "price": 130 }]
}
```

3.6 JSON Comments

JSON doesn't support comments by default. However, there are some tricks we can use, such as adding an extra attribute or key that work as comments in a JSON object.

Consider following example. Here, key-value pair ("comments": "This is comment") serve as our comment.

```
{
  "book": {
    "title":  "Advanced Web Technologies",
    "price":  200,
    "comments":  "This is comment"
  }
}
```

```
}}
```

Consider following example. Here, attribute "comment_1" serve as comment.

```
{
  "book": {
    "title":   "Advanced Web Technologies",
    "comment_1":   "This is good book"
  }
}
}}
```

JSON with javascript Example

Following example demonstrate creation of JSON object "book" using javascript.

```
<html>
<head>
<h3>Demo Example: JSON with JavaScript</h3>
<script >
    var book = { "title" : "Advanced Web Technologies", "author"  : "jmp" };
    document.write("<br>");
    document.write("Title = " + book.title);
    document.write("<br>");
    document.write("Author = " + book.author);
</script>
</head>
<body>
</body>
</html>
```

Output:

Demo Example: JSON with JavaScript

Title = Advanced Web Technologies
Author = jmp

JSON object with javascript Example:

Following example demonstrate creation of JSON object "book" and array "authors" using javascript.

```
<html>
<head>
<h3>Demo Example: JSON array with JavaScript</h3>
<script >
    var book = { "title" : "Advanced Web Technologies",
        "authors"  : ["jmp","mv"] };
    document.write("<br>");
    document.write("Title = " + book.title);
    document.write("<br>");
    document.write("Author name -1 : " + book.authors[0]);
    document.write("<br>");
    document.write("Author name -2 : " + book.authors[1]);
</script>
</head>
<body>
</body>
</html>
```

Output

Demo Example: JSON array with JavaScript

Title = Advanced Web Technologies
Author name -1 : jmp
Author name -2 : mv

Questions:

1. Explain JSON
2. Write JSON features
3. Write similarity between JSON and XML
4. Write difference between JSON and XML
5. Write about JSON array.
6. Write about JSON multi-dimensional array.
7. Explain JSON array of objects.
8. How will you write comments in JSON? Explain with example.

MCQ:

1. JSON name-value pair is specify as

A. name : 'value'

B. name' = 'value'

C. name = "value"

D. "name" : "value"

Ans: D

2. JSON strings must be in

A. single quote

B. double quote

C. single quote or double quote

Ans:B

3. According given JSON notation, book is of type

{ "book": ["web", "web design", "advanced concept"] }

A. Not a valid JSON string

B. Array

C. Class

D. string

Ans:B

4. Which is not a JSON type?

A. number

B. date

C. Array

D. string

Ans:B

5. Which of these is true for the JSON standard?

A. It is an open standard

B. It is privately developed

C. It requires a license to use

Ans:A

6. What is the file extension of JSON? OR Which of the following extension in used to save a JSON file?

A. .jon

B. .jsn

C. .json

D. .js

Ans:C

7. What is the common usage for JSON?

A. To store information local y.

B. To send and receive bits of data.

C. To store information remotely.

Ans:B

8. Which is proper a JSON array?

A. { ' alphabet ' : {"x", "y", "z ' } }

B. { "alphabet" : ["x", "y", "z";] }

C. { " alphabet " : [x, y, z] }

D. { " alphabet " : ["x", "y", "z"] }

Ans: D

9. According given JSON notation, book is of type

{ "book": {"title": "Advanced Web Techno ogies", "price":200 } }

A. Object

B. Array

C. XML

D. Not a valid JSON string

Ans: A

10. What does JSON stand for? OR What is the full form of JSON?

A. JavaScript Object Nomenclature

B. JavaScript Objective Notation

C. JavaScript Object Notation

D. JavaScript Orientated Nomenclature

Ans:C

11. A disadvantage of JSON is that it requires the use of JavaScript. OR JSON requires the use of JavaScript. True or false?

A. False, JSON is language independent.

B. True, though all browsers have JavaScript enabled.

D. True, though JavaScript is readily available in browsers.

Ans: A

12. What kind of format is JSON?

A. A lightweight data-encoding framework.

B. A lightweight data-interchange format.

C. A lightweight data-encryption format.

D. A lightweight database framework.

Ans: B

13. In JSON arrays, the order of elements is always preserved.

A. False

B. True

C. can't say

Ans: B

14. _____ is supported as a JSON Value type.

A. Infiniti

B. Null

C. NaN

D. Undefined

Ans: B

15. A string object in JSON, ___ separate the string and the value.

A. A space

B. A semicolon

C. A comma

D. A colon

Ans: D

16) Which of the following isn't a JSON type?

A. String

B.Cbject

C.Date

D..Array

Ans: C.

17. JSON elements are separated by ____

A. comma

B. line break

C. semi-colon

D. white space

Ans: A

18. In this example, what is the value of book.authors[1]?

{ "book" : { "title" : " Advanced Web Technologies", "authors" : ["jmp", "mv"] } }

A. null

B. undefined

C. jmp

D. mv

Ans: D

19. In this example, what is the TYPE of book.authors?

{ "book" : { "title" : " advanced web technology ", "authors" : ["jmp", "mv"] } }

A. boolean

B. Object

C. Array

D. String

Ans: C

20. In this example, what is the TYPE of book.price?:

{ "book": {"title": "advanced web technology", "price":200 } }

A. Number

B. boolean

C. String

D. Array

Ans:A

Unit – 4

AJAX (Asynchronous JavaScript and XML)

4.1 AJAX introduction

- AJAX stands for Asynchronous JavaScript and XML
- It allows us to send and receive data asynchronously without reloading/refreshing the entire web page.
- AJAX allow:
  - Update a web page without reloading/refreshing entire page i.e. it is possible to update parts of a web page, without reloading the whole page.
  - Request/receive data from a server - after the page has loaded
  - To create fast and dynamic web pages.
  - creating faster and more interactive web applications
- AJAX applications are browser- independent.
- AJAX applications are platform-independent.
- There are number of web applications that are using AJAX technology, such as:
  - Google Maps,
  - Gmail,
  - Youtube,
  - Google Suggest
  - Facebook

4.2 Fundamentals of AJAX technology:

AJAX is group of inter-related technologies includes:

- HTML/XHTML and CSS: They are used to display content and style. It is used for browser base presentation.
- XML or JSON: It is used to carry data to and from server.
- XMLHttpRequest: It allows asynchronous communication between client and server.
- DOM : It is used for dynamic display and interaction with data.
- JavaScript: It bring all technologies together.

4.3 Web-Application Model

4.3.1 Synchronous (Classic Web-Application Model)

Classic or traditional Web application model is Synchronous. It uses start-stop-start-stop approach in communication with server. In order to responds to user action, browser discard current HTML page. Then request is sent to server. When server finished the processing request, response is returned to browser. In last, browsers refresh the page and displays newly created HTML page. The user cannot perform any task during this process and browser is unresponsive. i.e. A synchronous request blocks the client or keep the user waiting until operation is completes.

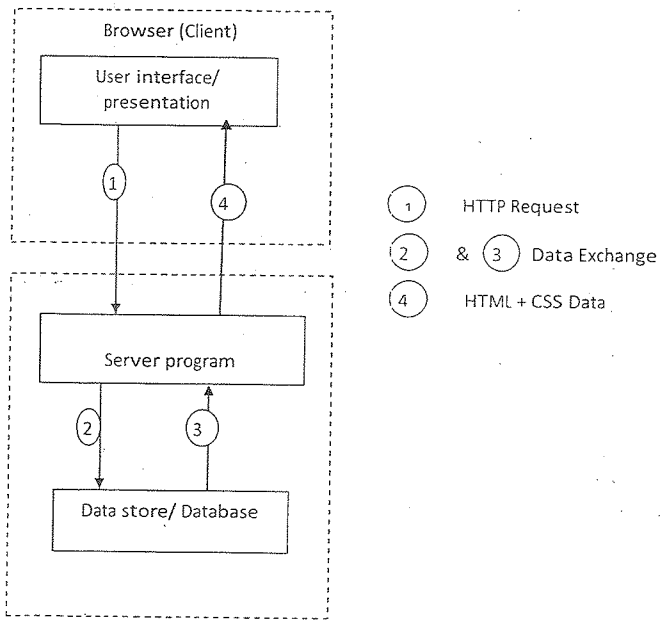


Figure 4.1: Classical model

Figure 4.1 show classical web application model.

Figure 4.1 shows that the full page is refreshed at request time. The user is blocked until the request is completed.

4.3.2 Asynchronous (AJAX Web-Application Model)

AJAX web application model is Asynchronous. It doesn't uses start-stop-start-stop approach in communication with server. It uses AJAX engine. AJAX engine is written in JavaScript. AJAX engine perform interaction between user interface and AJAX engine on the client side. An asynchronous interaction doesn't block the client i.e. browser is responsive. At that time, user can perform another operation or interact with browser.

Figure 4.2 shows that the entire page is not refreshed at request time. User gets response from the Ajax engine.

4.4 XMLHttpRequest (XHR) Object

- XMLHttpRequest (XHR) object is the key of AJAX.
- It is used to interact or exchange data with servers. This is an asynchronous communication. As a result, it is possible to update portions of a web page without reloading the entire page.
- It is supported by all modern browsers.
- It can be used with JavaScript, VBScript, and other scripting languages to transfer and manipulate XML data. It can be used to retrieve data in variety of formats, such as plain text, JSON, and so on.

XMLHttpRequest performs following tasks:

1. Sends data from the client to server in the background
2. Receives the data from the server
3. Updates the webpage without having to reload it.

Create an XMLHttpRequest Object

To create XMLHttpRequest object use following syntax

variable = new XMLHttpRequest();

4.4.1 Properties of XMLHttpRequest Object

XMLHttpRequest Object support following properties:

(i). **onreadystatechange**:

onreadystatechange specify a name of function or event handler to be called automatically when readyState property changes.

(ii). **readyState**:

It represent or holds current the status of the XMLHttpRequest. It's possible value is from 0 to 4 as per following:

value	state	Explanation
0	UNOPEN	It indicates request is not initialized. i.e. open() is notcalled
1	OPENED	It indicates the request has been set up. i.e. open is called but send() is not called.
2	HEADERS_RECEIVED	It indicates send() is called, and headers and status are available.
3	LOADING	It indicates the request is in process. Downloading data; *responseText* holds the data.
4	DONE	It indicates request finished/completed and response is ready

(iii). **responseText:**

It is read only. It returns the response data/text.

(iv). **responseXML:**

It is read only. It returns the response data as XML data.

4.4.2 Methods of XMLHttpRequest Object

XMLHttpRequest Object support following methods:

(i). open() :

open() method is used to open or Initializes an HTTP request for sending.

Syntax: open(method, url [, async[, uname, pswd]])

Parameters:

parameter	description
method	It specify the type of request- GET or POST
url	It specify the location of the file on the server
async	It specify whether request is asynchronous (default) or not. If it is set to true then request is asynchronous. If it is set to false then request is synchronous.
uname	it is Optional. It specifies the name of the user for authentication.
Pwd	it is Optional. It specifies the password of the user for authentication.

(ii). send():

Send () method is used to send the request to the server.

Syntax: send([string])

string: data to be sent in the request. It is used when request is send to the server using POST. If the request method is GET, than string parameter is ignored and the request string is set to *null*.

(iii). setRequestHeader():

setRequestHeader() method adds a header/value pair to the HTTP request header to be sent. It must call after open(), but before calling send().

Syntax: setRequestHeader(header,value)

Header : The name of the header whose value is to be set.

Value: The value to set as the body of the header.

4.5 Working of AJAX and its architecture

AJAX use XMLHttpRequest object to communicate with the server. The figure 4.2 illustrates the flow or working of AJAX. In figure, following symbol notation are used:

(1) JavaScript call

(2) HTTP Request

(3) & (4) Data Exchange

(5) XML or JSON Data

(6) HTML + CSS Data

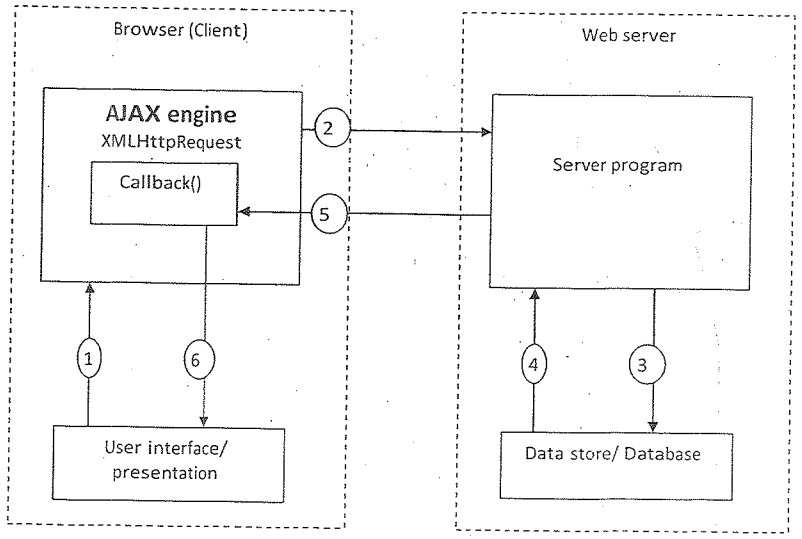


Figure 4.2: Working of AJAX

AJAX create an instance of XMLHttpRequest object and send an HttpRequest request to the server. The server process the request, send data back to the browser (or client). The web browser process data using JavaScript and updates the data of web page. The following steps show flow of AJAX.

1. When user sends a request, a JavaScript create XMLHttpRequest object.
2. XMLHttpRequest object create HTTP Request (HttpRequest) and sent to the server using send() metod..
3. The server processes the request and interacts with the database.
4. Data is retrieved
5. Server sends XML data or JSON data to the XMLHttpRequest callback function. i.e. server sends a response back to the web page.
6. The response is read, proper action (like page update, display data on browser) is performed.

Questions:

1. What is AJAX?
2. Write features of AJAX.
3. Write difference between Synchronous and Asynchronous web application.
4. Explain XMLHttpRequest objects.
5. Explain various properties of XMLHttpRequest.
6. Explain various methods of XMLHttpRequest.
7. Explain working of AJAX.

MCQ

1.Ajax stands for Synchronous JavaScript and XML.

A.True

B.False

Ans:B False

2. AJAX Stands for:

A. Asynchronous Javascript and XML

B. Asynchronous JSON and XML

C. Asynchronous Java Abstraction for X-Windows

D. Another Java and XML Library

Ans: A. Asynchronous Javascript and XML

3. Which of the following are the features of Ajax?

A. Live data binding B. Declarative instantiation of client components

C. Client-side template rendering

D. All of the above

Ans: D. All of the above

4. The advantage(s) of Ajax is (are)

A. Bandwidth utilization B. More interactive C. Speeder retrieval of data

D. All of given

Ans. D. All of given

5. What sever support AJAX ?

A. WWW B. SMTP C. FTP D. HTTP

Ans: D HTTP

6. can AJAX work with web application ?

A. true B. False C. cant's say

Ans: A. true

7. The XMLHttpRequest object with regards to Ajax...

A. It is the language used to create Ajax applications.

B. It enables the exchange of structured data between the Web server and the client.

C. It enables asynchronous data exchange between Web browsers and a Web server.

D. It allows you to mark up and style the display of Web-page text.

Ans: C It enables asynchronous data exchange between Web browsers and a Web server.

8. _____combination of technologies gives AJAX its name.

A. Asynchronous Computing and DHTML

B. Asynchronous JavaScript and XML

C. ASP and XAML

D. Alias and XML

Ans: B Asynchronous JavaScript and XML

9. Which technologies is not used in AJAX?

A. CSS

B. DHTML

C. DOM

D. Webex

Ans: D Webex

10. AJAX is based on

A. JavaScript and XML

B. Java and JavaScript

C. java and XML

D. JavaScript and HTTP requests

Ans: A JavaScript and XML

11. Which property defines a function to be executed when the readyState changes?

A. onreadystatechange

B. onreadystatedown

C. onreadystateup

D. None of the above

Ans: A onreadystatechange

12. The technologies used by Ajax is(are)...

A. JavaScript

B. XMLHttpRequest

C. (DOM) Document Object Mode

D. All of the above

Ans : D All of the above

13. __ is a technique for creating fast and dynamic web pages.

A. AJAX

B. XML

C. JSON

D. All of the above

Ans: A AJAX

14. The purpose of XMLHttpRequest is ..

A. Load content by loading new document

B. Multiple loading

C. Load content without loading new document

D. None of the mentioned

Ans: C Load content without loading new document

15. The purpose of XMLHttpRequest is ..

A. Load content by loading new document

B. Multiple loading

C. Load content without refreshing the page

D. None of the mentioned

Ans: C Load content without refreshing the page

16. Which method of XMLHttpRequest is used to Initializes an HTTP request for sending?

A. open

B. send

c. init

d. request

Ans: A. open

17. Respect to XMLHttpRequest, Which method must call after open(), but before calling send() ?

A. setRequestHeader()

B. request()

C. setRequest()

D. none

Ans: A setRequestHeader()

18. _____ technology provides the ability to dynamically interact with Web page layout.

A. XML

B. JavaScript

C. Document Object Model.

D. HTML.

Ans: C Document Object Model

19 The XMLHttpRequest object is used in Ajax to exchange data with the web server.

A. true

B. false

Ans: A true

Unit - 5

Node.js

5.1 Concept, Working and Features

- Node.js is a server-side platform. It is built on Google Chrome's JavaScript Engine (V8 Engine).
- It was developed by Ryan Dahl in 2009 and its latest version is 16.13.1 (includes npm 8.1.2)
- The definition of Node.js given by its official documentation is as follows –
- "Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices."
- Node.js is an open source, cross-platform runtime environment.
- It is used for developing server-side and networking applications.
- Node.js applications are written in JavaScript.
- It can be run inside the Node.js runtime on OS X, Microsoft Windows, and Linux.
- Node.js also provides a huge library of different JavaScript modules
- Such libraries make the development of web applications easier.

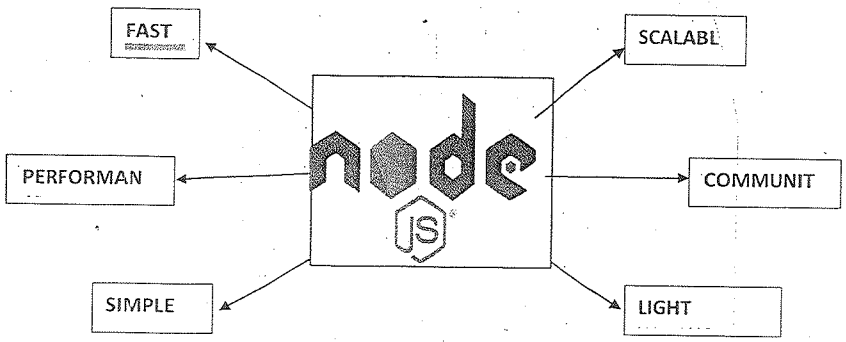
Node.js = Runtime Environment + JavaScript Library

5.1.1 Features of Node.js

Following are some of the important features of Node.js:

- **Asynchronous** – All APIs of Node.js library is asynchronous. This means that they are non-blocking. Server based on Node.js never waits for an API to return data. The server moves to the next API after calling it.
- **Event Driven** The event notification of Node.js helps the server to get a response from the previous API call.
- **Very Fast** –Since it is built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded** - Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way.

- **Highly Scalable**- Node.js uses a single threaded program and the same program can provide service to many requests making it highly scalable
- **No Buffering** –Applications of Node.js never buffer any data. They just give data as output in chunks.
- **License** – Node.js is released under the MIT license.



Following are the areas or applications where Node.js can preferred to be used:

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Where Not to Use Node.js?

It is not advisable to use Node.js for CPU intensive applications.

5.1.2 Downloading Node.js

- Node.js can be downloaded from its official website https://nodejs.org/en/download/.
- On this site click the **Windows Installer** button to download the latest default version.
- The Node.js installer also includes the NPM. NPM is **the package manager for the Node JavaScript platform**. It puts modules in the location from where the node can

find them, and use them whenever required. So in general, it is used to publish, discover, install, and develop node programs.

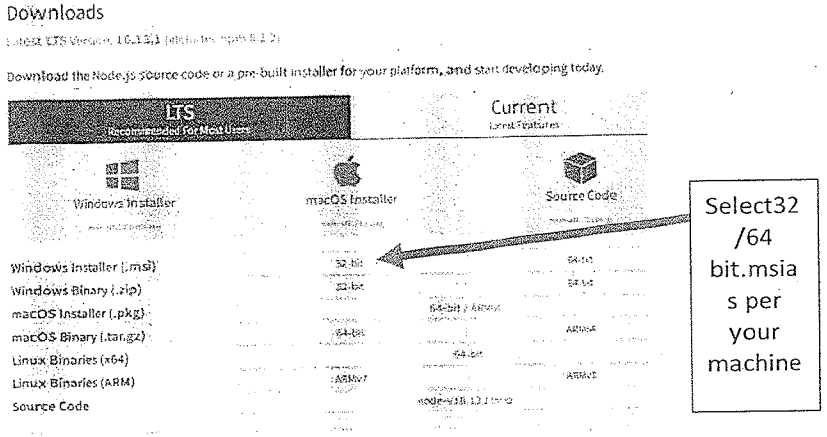
5.2 Setting up Node.js server

5.2.1 Installing on window

Below are the steps to download and install Node.js in Windows:

Step 1) Download Node.js Installer for Windows

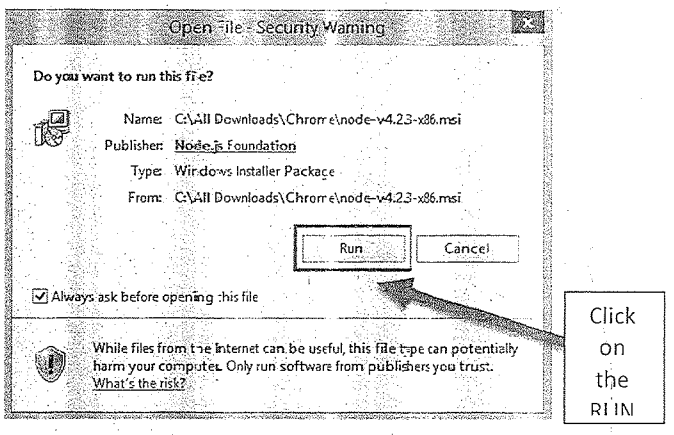
Go to the site https://nodejs.org/en/download/ and download the necessary binary files.



Step 2) Run the installation

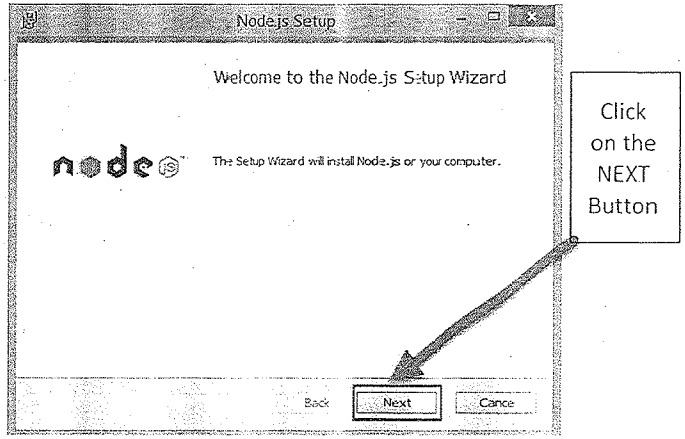
Double click on the downloaded .msi file to start the installation.

In the first screen ,click the Run button to begin the installation.



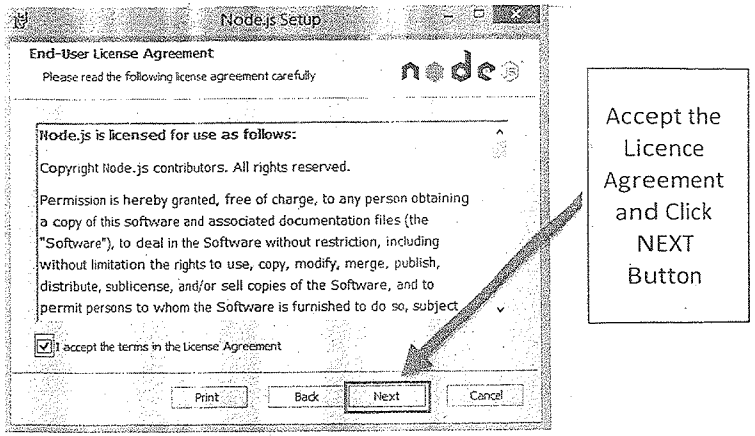
Step 3) Continue with the installation steps

In the next screen, click the "Next" button to continue the installation



Step 4) Accept the terms and conditions

In the next screen, Accept the license agreement and click on the Next button.

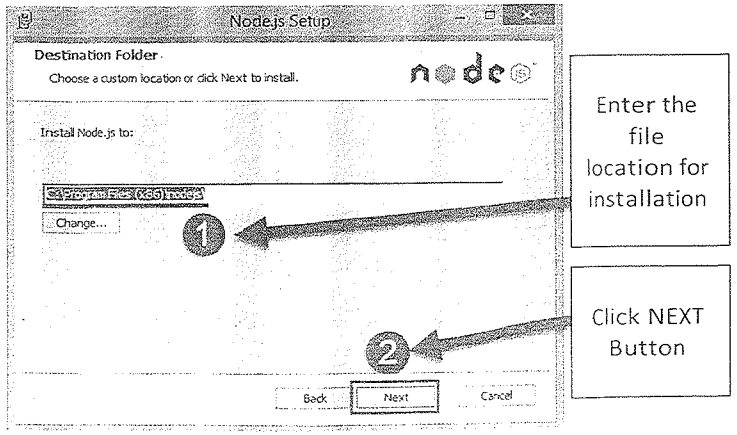


Step 5) Set up the path

In the next screen, choose the location where Node.js needs to be installed and then click on the Next button.

1. First, enter the file location for the installation of Node.js. This is where the files for Node.js will be stored after the installation.

2. Click on the Next button to proceed ahead with the installation.

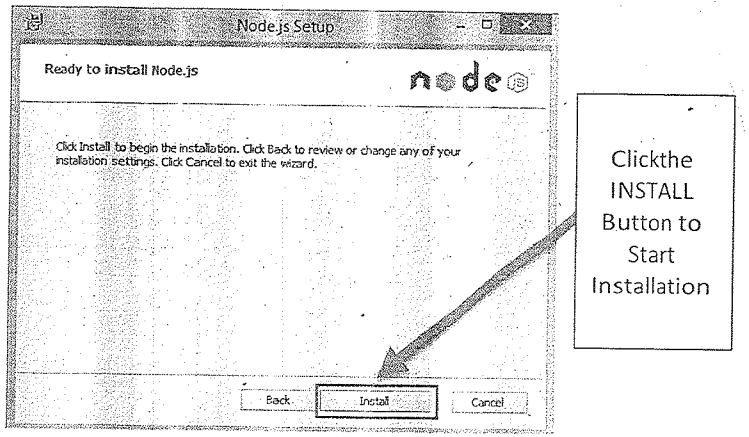


Step 6) Select the default components to be installed

Accept the default components and click on the Next button.

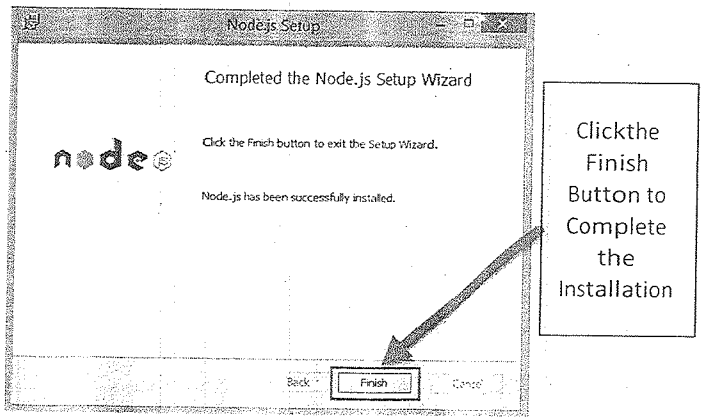
Step 7) Start the installation

In the next screen, click the Install button to start installing Node.js on Windows.



Step 8) Complete the installation

Click the Finish button to complete the installation.

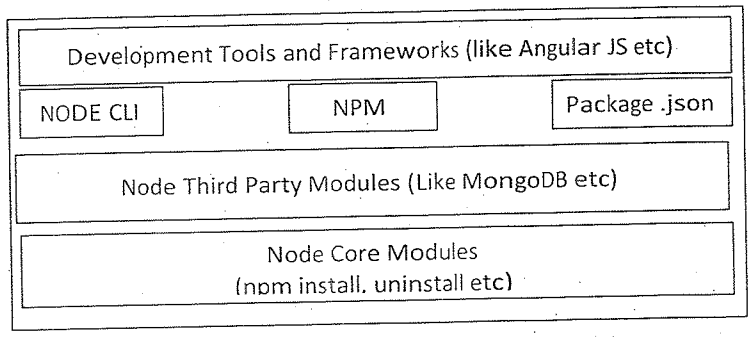


5.2.2 Components

- Node JS contains many components to develop, test and deploy applications.
- Following are the list of Node JS components:
  - Node CLI

- NPM
- package.json
- Node Modules
- Development Tools and Frameworks

The following figure depicts **NODE JS PLATFORM**:



5.2.2.1 Node CLI

Node JS has a CLI (Command Line Interface) to run the basic commands and script files.

This component gets installed by default while installing Node JS Platform.

We can open command prompt and type basic JavaScript commands or even files on the command prompt

Node.js command prompt - node.exe

```
Your environment has been set up for using Node.js 16.13.0 (x64) and npm.

C:\Users\User>node.exe
Welcome to Node.js v16.13.0.
Type ".help" for more information.
>
```

5.2.2.2 NPM

NPM stands for Node Package Manager. NPM is used to install, update, uninstall and configure Node JS modules/packages very easily.

When we install Node JS Base Platform, it installs only few components, modules and libraries like Node CLI, NPM etc. Later on, we can use NPM to upgrade Node JS with our required modules.

To check npm version, run "**npm -v** command on Node CLI :.

Node.js command prompt

```
C:\Users\User>npm -v
8.1.0

C:\Users\User>
```

5.2.2.3 package.json

"**package.json**" is a plain text file in JSON format. It is used to manage our application required module dependencies. We should place this file in our application root folder.

It defines information like our application name, module dependencies, module versions etc. This configurations file is very important and requires more time to explain in detail. We will discuss it in detail with some examples in coming posts.

Myownapp package.json file;

```
{
 "name" : "myownapp",
 "version" : "1.0",
 "dependencies" : {
 }
}
```

5.2.2.4 Node Modules

Node JS is a modular platform. Each functionality of it is implemented by a separate module or package. It has some core modules like npm, install, uninstall, update etc and rest all modules are third-party modules.

When we install Node JS Platform, by default only one module is installed i.e.npm module.

We need to use "npm" command to install required modules one by one.

All Core or Default modules are installed at /lib folder as *.js files.

Node JS has thousands of modules. A full list of Node JS Platform's packages or modules can be found on the NPM website https://npmjs.org/.

5.2.2.5 Development Tools and Frameworks

Many companies have developed some tools and framework to ease and reduce the overhead of Node JS applications since the Node JS Platform has become very popular in developing Data-Sensitive Real-time and Network applications.

5.2.3 Request and response object

A request (an http.IncomingMessage object) and a response (an http.ServerResponse object) objects are the parameters of the callback function and are essential to handle the HTTP call.

Let us discuss each object in detail.

5.2.3.1 Request Object

The Request object is used to get information from a client and send to the server. It contains properties for the request query string, HTTP headers, parameters, body, etc. The properties of Request object, req, can be accessed through a dot(.) operator as req.property.

Few of itsproperties are listed as follows:

Property	description
baseurl	Denote the URL on which a router instance was mounted.
body	Hold key-value pairs of data submitted in the request body.
cookies	Used to read cookies data.
hostname	Specifies the hostname from the http header.
path	Specifies the path part of the request URL.

fresh	Specifies that the request is "fresh." it is the opposite of stale.
ip	Specifies the remote IP address of the request.
originalurl	Retains the original request URL, also allows to rewrite url for internal routing purposes.
params	Contain properties mapped to the named routeparameters. For example, if you have the route as /user/:name, then the "name" property is available as req.params.name. This object defaults to {}.
protocol	The request protocol string, "http" or "https" when requested with TLS. (Transport Layer Security)
query	Contains a property for each query string parameter in the route.
route	The currently-matched route, a string.
secure	A Boolean that is true if a TLS connection is established.
stale	It indicates whether the request is "stale," and is the opposite of fresh property.
subdomains	It represents an array of subcomains in the domain name of the request.

5.2.3.2 Response Object

The Response object is used to send information from server to the client. i.e. the response object is going to client as response from the server. The methods of Response object, res, can be accessed through a dot(.) operator as res.method().

Following are few important methods of Request object.

Method name	Description
end()	Used to end the response process. It signals the server that the response is complete.
append()	Used to append data to the HTTP response header field.
attachment()	Used to send a file as an attachment in the HTTP response
cookie()	Used to set cookie name to value
format()	Used to content negotiation on the Accept HTTP header
get()	Used to provides HTTP response header specified by field.
set()	Set the response's HTTP header field .

json()	Returns the response in JSON format.
jsonp()	Used to returns the response in JSON format with JSONP support
location()	Based on the path parameter, it sets the response location HTTP header field.
links()	Used to set the response's Link HTTP header field by means of joining the links provided as properties of the parameter.
redirect()	Redirect request to given URL.
send()	Used to send the HTTP response.
sendStatus()	Used to set the response HTTP status code
write()	Writes text to client, i.e. write text in browser.
writeHead()	Sends status and response headers to the client
type()	Set the Content-Type HTTP header to the MIME type

5.3 Built-in modules

- Module in Node.js is a simple or complex functionality organized in single or multiple JavaScript files
- These modules can be reused throughout the Node.js application.
- Each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope. Also, each module can be placed in a separate .js file under a separate folder.
- Node.js implements CommonJS modules standard.
- CommonJS is a group of helpers who define JavaScript standards for desktop, web server and console application.

Node.js Module Types

Node.js includes three types of modules:

1. Core Modules
2. Local Modules(user-defined modules)
3. Third Party Modules

5.3.1 Core Modules: Node.js has many built-in modules. They are part of the platform and comes along with Node.js installation. These modules can be loaded into the program by using the **require** function.

Syntax:

var module = require('module_name');

The *require()* function will return a JavaScript type depending on what the particular module returns. The following example demonstrates how to use the Node.js Http module to create a web server.

```
varhttp = require('http');
http.createServer(function(req, res) {
 res.writeHead(200, {'Content-Type': 'text/html'});
 res.write("Advanced Web Technology!');
 res.end();}).listen(3000);
```

In the above example, the require() function returns an object because the Http module returns its functionality as an object. The function http.createServer() method will be executed when someone tries to access the computer on port 3000. The res.writeHead() method is the status code where 200 means it is OK, while the second argument is an object containing the response headers.

The following list contains some of the important core modules in Node.js:

Core Modules	Description
http	creates an HTTP server in Node.js.
fs	used to handle file system.
path	includes methods to deal with file paths.
querystring	used for parsing and formatting URL query strings.

os	provides information about the operating system.
process	provides information and control about the current Node.js
url	provides utilities for URL resolution and parsing.
assert	set of assertion functions useful for testing.

5.3.2 Local Modules(User Defined Modules):

Local modules are created locally in your Node.js application. You can create your own modules, and easily include them in your applications.

The following example demonstrates how to create a user-defined module and include it in our node.

Example:

Create a calc.js file that has the following code:

```
exports.add= function(x, y) {
  returnx + y;
};
  exports.sub = function(x, y) {
  returnx - y;
};
 exports.mult = function(x, y) {
  returnx * y;
};
  exports.div = function(x, y) {
  returnx / y;
};
```


The *exports* keyword is used to make properties and methods available outside the module file. Since this file provides attributes to the outer world via exports, another file can use its exported functionality using the *require()* function.

Now you can include and use the module in any of your Node.js files. Let's create a file with name implement_calc.js

Filename: implement_calc.js

```
varcalculator = require('./calc');
varx = 30, y = 10;
console.log("Addition of 30 and 10 is "
        + calculator.add(x, y));
 console.log("Subtraction of 30 and 10 is "
        + calculator.sub(x, y));
console.log("Multiplication of 30 and 10 is "
        + calculator.mult(x, y));
console.log("Division of 30 and 10 is "
        + calculator.div(x, y));
```

The use ./ to locate the module, that means that the module is located in the same folder as the implement_calc.js file.

Save the code above in a file called "implement_calc.js", and initiate the file:

```
node implement_calc.js
```

Output:

```
C:\Users\User\node>node implement_calc.js
Addition of 30 and 10 is 40
Subtraction of 30 and 10 is 20
Multiplication of 30 and 10 is 300
Division of 30 and 10 is 3

C:\Users\User\node>
```

Note: This module also hides functionality that is not needed outside of the module.

5.3.3 Third-party modules: Third-party modules are modules that are available online using the Node Package Manager(NPM). These modules can be installed in the project folder or globally. Some of the popular third-party modules are mongoose, express, angular, and react.

Example:

- npm install express
- npm install mongoose
- npm install -g @angular/cli

5.3.4 HTTP Module

- Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).
- To include the HTTP module, use the *require()* method:

  var http = require('http');
- The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.
- The *createServer()* method is used to create an HTTP server:

Example:

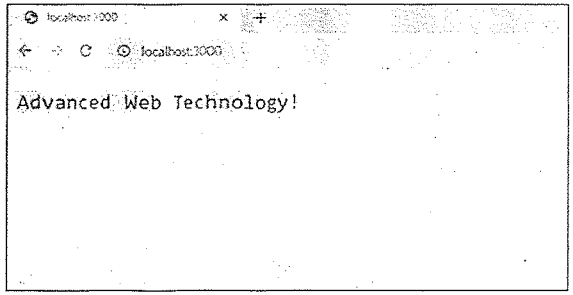
```
var http = require('http');
//create a server object:
```



```
http.createServer(function (req, res) {
res.write('Advanced Web Technology!'); //write a response to the client
res.end(); //end the response
}).listen(3000); //the server object listens on port 3000
```

- The function passed into the http.createServer() method, will be executed when someone tries to access the computer on port 3000.
- Save the code above in a file called "http_demo.js", and initiate the file:
- Run demo_http.js: using node demo_http.js on command prompt.
- If you type http://localhost:3000in URL bar of any browser, you will get following output:

Output:



Note: This module will be discussed in detail in next section

5.4 Node.js as a Web Server

Web server is required to access the web pages of any web application. The web server handles all the http requests coming from the web application. For example, Apache web server is used for PHP web applications, and IIS web server is used for ASP.NET web applications.

Node.js allows to create own web server that handle incoming HTTP requests asynchronously.Node.js has a built-in module called HTTP. This module enables Node.js to send data over HTTP. The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

Methods:

method	description
createServer()	create HTTP server
writeHead()	Add HTTP header to the request

5.4.1 Creating Web Server and AddingHTTP header:

5.4.1.1 createServer() method:

createServer() is used to create HTTP server. It returns an HTTP Server object.This method includes a request object, which is used to obtain information about the current HTTP request, such as request header, the url, and data.

Syntax

http.createServer([options][, requestListener]);

Where, parameters are as follows:

(i) options:

options	description	Default value
IncomingMessage	It specifies the IncomingMessage class to be used.	IncomingMessage
ServerResponse	It specifies the ServerResponse class to be used.	ServerResponse
insecureHTTPParser	When it is set to true, It use an insecure HTTP parser that accepts invalid HTTP headers.	false
maxHeaderSize	It specifies the maximum length of request headers in bytes.	16384 (16 KB).

(ii) requestListener: This parameter is optional. It specifies a function that will be called whenever the server receives a request. This function is known as a *requestListener*.requestListenerhandles user requests andgives response back to the user.

5.4.1.2 writeHead() method:

writeHead() method write a header to the response. It is also used to specify the status code and the content type.

Syntax:

writeHead(status_code[, status_message][, headers]);

where

status_code: It is a 3-digit HTTP status code. For example, 404 for file not found, 200 for ok, 505 for Internal Server Error.

status_message: It is string that represents the status message

headers: It takes any function, string or array.

Return value: It returns a reference to the ServerResponse, so that calls can be chained

For example: Following code set content header. Here, first argument is status code 200, means all is "ok". Second argument is object which represent type of contain.

writeHead(200, {'Content-Type': 'text/html'});

5.4.1.3 How to create web server

Use createServer() to create web server.

The following example demonstrates how to use Node.js http module to create a web server.This code creates a server that will listen on port number 8000. If a browser request is made on this port number, the server will respond to the client.The function passed in thecreateServer() will be executed when the client visitthe url http://localhost:8000

```
var http = require('http');                    // Import http module

//create a server object:                      // create a server object and call
var server= http.createServer(function(req, res) function with request and
                                               response object
{
// code                                        // write code here
}
```

```
);
server.listen(8000);                    //Listen to port number 8000
```

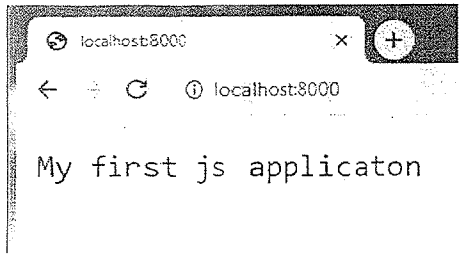
Example: The following code create web server and display message "'My first js application'" in browser.

```
var http = require('http');  //import http module
        //create a server object:
http.createServer(function (req, res)
{
        res.write('My first js application'); //write a response to the client
        res.end(); //ending the response
}).listen (8000); // server is set to listen on the 8000 port
```

Steps to run the code:

1. Save the code in a file with .js extension, say *test.js*.
2. Open the command prompt
3. Using the *cd* command, navigate to the folder containing the file.
4. Run the command: node *file_name.js* . Here, write node*test.js*
5. Open the browser
6. Type URL http://localhost:8000 in URL bar.

Output: In browser, output will be look like:



Above example can be written as

```
var http = require('http');
var server = http.createServer(function (req, res)
{
        res.write('My first jsapplicaton');
        res.end();
});
server.listen(8000);
```

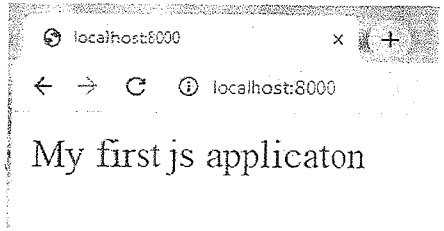
5.4.1.4 How to add a HTTP Header

Use writeHead() method to add header.

Example: Following code add http header.

```
var http = require('http');
//create a server object:
http.createServer(function (req, res)
{
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.write('My first jsapplicaton'); //write a response to the client
        res.end(); //end the response
}).listen(8000); //the server object listens on port 8080
```

Output :



5.4.2Query String

Query strings are appended to the end of a URL followed by ?. The query string contains parameters in the form of a key-value pair. The key-value pare is separated by by the equal (=) symbol. The ampersand (&) symbol separates each parameter.

For example,

(1) https://jump2learn.com?book=advanced // here book is key and advanced is value

(2) https://jump2learn.com?book=advanced&price=200 // it contains two parameters

5.4.2.1 Reading Query string data

The function, which is passed into the createServer(), has first argument as request object. This request object represents the request from the client.The Request object is used to get information from a client and send information to server. The request object has properties for the request query string, body, parameters, body, HTTP headers etc.

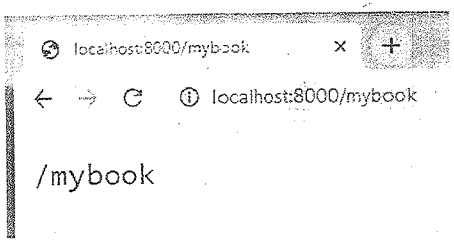
"url" property of request object contains the portion of the url that follows the domain name. For example, if user type https://jump2learn.com/userguide in browser, then the value of *url* property is "/userguide"

Example: consider following code.

```
var http = require('http');
http.createServer(function (req, res)
{
        res.write(req.url);
        res.end();
}).listen(8000)
```

Output:

If user type http://localhost:8000/mybook in the URL bar, browser will display:



5.4.2.2 Splitting Query string

Splitting of query string data means parse or separate query string parameter, so they can process individually. To split query string data we need to first import "url" module, use *url.parse* method and query property.

url.parse():

The url.parse() method parses a URL string. It split various elements of the URL such as host, pathname, search keys, etc.

Syntax:

url.parse(urlString[, parseQueryString])

where,

urlString: The URL string to parse.

parseQueryString: It is Boolean value. It indicates whether the method should parse the query string. If it is true, it parses query string. If it is false, it will return URL object unparsed, undecoded string. Its Default value is false.

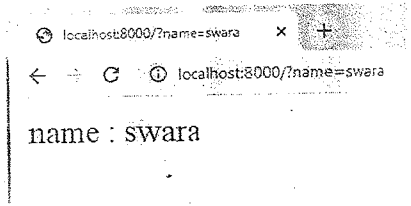
query:

.query property:It rerun object containing a property for each query string parameter.

Example: Following code read query string data "name".

```
var http = require('http');
var url = require('url');              //import url module
http.createServer(function (req, res)
{
        res.writeHead(200, {'Content-Type': 'text/html'});
          var query_data = url.parse(req.url, true).query;
        res.write("name : " + query_data.name);
 }).listen(8000);
```

Output: If you type http://localhost:8000/?name=swara in URL bar, it will display:



Explanation:

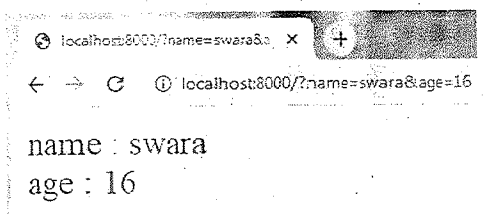
- Script line var url = require('url') importurl module and return reference to "url" variable.
- Script line var query_data = url.parse(req.url, true).query; is important. Here,
  i. *req.url* will hold/?name=swara.
  ii. url.parsesplit the various elements of the *req.url.*
  iii. .query property return query string data.
- *query_data.name* refer to "name" key in query string.

Reading multiple data

Following code parse multiple query string data.

```
var http = require('http');
var url = require('url');
http.createServer(function (req, res)
{
        res.writeHead(200, {'Content-Type': 'text/html'});
          var query_data = url.parse(req.url, true).query;
        res.write("name : " + query_data.name);
        res.write("<br/>");
        res.write("age : " + query_data.age);
        res.end();
}).listen(8000);
```


Output: If you type http://localhost:8000/?name=swara&age=16 in URL bar, it will display:



5.5 File System modules

- The Node.js file system module enables us to work with the file system on one's own computer.
- To include the File System module, we use the *require()* method as below:
- var fs = require('fs');
- Every method in the 'fs' module has synchronous as well as asynchronous forms.
- Asynchronous methods take the last parameter as the completion function callback and the first parameter of the callback function as error.
- It is better to use an asynchronous method instead of a synchronous method,
- As the asynchronous method never blocks a program during its execution, whereas the second one does.
- Some common uses of 'fs' module are:
  - Read Files
  - Create Files
  - Update Files
  - Delete Files
  - Rename Files
- To demonstrate the above uses let us first create a simple text file and save it as test.txt. Consider that the test.txt has following content:

  This is a test file for demonstrating the use of different uses of file system modules of node js in the book Advanced Web Technology!!

5.5.1 Read Files

- The *fs.readFile()* method is used to read files on one's own computer.
- Let us create a js file named read_demo.js with the following code:

```
var fs = require("fs");

// Asynchronous read
fs.readFile('test.txt', function (err, data) {
 if (err) {
  return console.error(err);
 }
console.log("Asynchronous read: " + data.toString());
});
// Synchronous read
var data = fs.readFileSync(test.txt');
console.log("Synchronous read: " + data.toString());
console.log("Program Completed");
```

- Run the above code as follows:

```
node read_demo.js
```

Output:

```
C:\Users\User\node>node read_demo.js
Synchronous read: This is a test file for demonstrating the use of
different uses of file system modules of node js
in the book Advanced Web Technology!!
Program Completed
Asynchronous read: This is a test file for demonstrating the use of
different uses of file system modules of node js
in the book Advanced Web Technology!!
```

5.5.2 Create Files

- The File System module of Node js has following methods for creating new files:

  - fs.appencFile()
  - fs.open()
  - fs.writeFile()

5.5.2.1 Opening a file

- fs.open() method is used to open a file for different purpose depending on the flag.
- Following is the syntax of the method to open a file in asynchronous mode–

  fs.open(path, flags[, mode], callback)
- In the above syntax :
  - path – filename including path that needs to be opened.
  - flags – behavior of the file to be opened. All possible values have been listed below.
  - mode – It sets the file mode (permission), but only when the file was created. It defaults to 0666, readable and writeable.
  - callback – This is the callback function which gets two arguments (err,fd). err- error while performing any operation on file and fd is file data
- Flags for read/write operations are –

Flag.	Description (Open file for)
r	Reading.
r+	Reading and writing.
rs	Reading in synchronous mode.
rs+	Reading and writing, asking the OS to open it synchronously.
w	Writing. The file is created (if it does not exist) or truncated (if it exists).
wx	Like 'w' but fails if the path exists.
w+	Reading and writing. The file is created (if it does not exist) or truncated
wx+	Like 'w+' but fails if path exists.
a	Appending. The file is created if it does not exist.
ax	Like 'a' but fails if the path exists.

a+	Reading and appending. The file is created if it does not exist.
ax+	Like 'a+' but fails if the path exists.

Example

The following code shows how to open a file test.txt for reading and writing.Save this file as open_demo.js.

```
var fs = require("fs");
// Asynchronous - Opening File
console.log("Trying to open file!");
fs.open('test.txt', 'r+', function(err, fd) {
  if (err) {
    return console.error(err);
  }
console.log("File opened successfully!");
});
```

Run the open_demo.jsto get the following output:

```
$ node open_demo.js
```

Output:

```
C:\Users\User\node>node open_demo.js
Trying to open file!
File opened successfully!
```

5.5.2.2 Writing in a file:

- *fs.writeFile()* method is used to create a new file if it does not exist. It will over-write the file if the file already exists.
- Syntax:

```
fs.writeFile(filename, data[, options], callback)
```

  - path – string having the filename including path.

  - data –The string or buffer to be written into the file.
  - options – It's an object which will hold {encoding, mode, flag}. By default. encoding is utf8, mode is octal value 0666. and flag is 'w'.
  - callback – This is the callback function which gets a single parameter err that returns an error in case of any writing error.

Example

The following code shows how to create and write in new file named test2.txt for reading and writing.Save this file as write_demo.js.

```
var fs = require("fs");
console.log("Trying to write in a new file");
fs.writeFile('test2.txt', 'Easy Learning! With Advanced Web Technology! book',
function(err) {
  if (err) {
    return console.error(err);
  }
console.log("Data written successfully!");
console.log("Let's read newly written data");
fs.readFile('test2.txt', function (err, data) {
    if (err) {
      return console.error(err);
    }
console.log("Asynchronous read: " + data.toString());
  });
});
```

Output:

```
C:\Users\User\node>node write_demo.js
Trying to write in a new file
Data written successfully!
Let's read newly written data
Asynchronous read: Easy Learning! With Advanced Web Technology! book
```

5.5.2.3 Appending in a file

- *fs.appendFile()*method is also used to create a new file if it does not exist. It will append the content in the file if it already exists.
- Syntax:
  - fs.appendFile(filename, data[, options], callback)
- path – string having the filename including path.
- data –The string or buffer to be written into the file.
- options – It's an object which will hold {encoding, mode, flag}. By default. encoding is utf8, mode is octal value 0666. and flag is 'a'.
- callback – This is the callback function which gets a single parameter err that returns an error in case of any writing error.
- Example
- The following code shows how to create and write in new file named test3.txt for reading and creating file using appendFile() method of fs.Save this file as append2create_demo.js.
  - var fs = require("fs");

```
console.log("Trying to create a new file with appenFile() method of File System");
fs.appendFile('test3.txt', 'Advanced Web Technology book by Jump2Learn Publishing! ',
function(err) {
        if (err) {
          return console.error(err);
        }
      console.log("Data written using append method successfully!");
      console.log("Let's read newly written data");
      fs.readFile('test3.txt', function (err, data) {
          if (err) {
            return console.error(err);
          }
      console.log("Asynchronous read: " + data.toString());
```



```
      });
});
```

Output:

```
C:\Users\User\node>node append2create_demo.js
Trying to create a new file with appenFile() method of File System
Data written using append method successfully!
Let's read newly written data
Asynchronous read: Advanced Web Technology book by Jump2Learn Publishing!
```

5.5.3 Update Files

- The File System module has methods for updating files:
  - fs.appendFile()
  - fs.writeFile()
- The fs.appendFile() method appends the specified content at the end of the specified file:
- Example
- The following code append the text "Very Informative and Good Book." to the end of the file "text3.txt".Save this file as append2update_demo.js.

```
var fs = require("fs");
console.log("Trying to update an existing file with appenFile() method of File
System");
fs.appendFile('test3.txt', 'Very Informative and Good Book! ', function(err) {
  if (err) {
    return console.error(err);
  }
console.log("Data appended using append method successfully!");
console.log("Let's read newly updated data");
fs.readFile('test3.txt', function (err, data) {
   if (err) {
     return console.error(err);
   }
```

```
    console.log("Asynchronous read: " + data.toString());
  });
});
```

Output:

```
C:\Users\User\node>node append2update_demo.js
Trying to update an existing file with appendFile() method of File System
Data appended using append method successfully!
Let's read newly updated data
Asynchronous read: Advanced Web Technology book by Jump2Learn Publishing! Very Informative and Good Books
```

- The fs.writeFile() method replaces the specified file and its content.
- **Example:**
- The following code overwrites the content of the file "text3.txt".Save this file as write2update_demo.js.

```
var fs = require("fs");
console.log("Trying to overwrite an existing file with writeFile() method of File System");
fs.writeFile('test3.txt', 'New Overwritten text! ', function(err) {
  if (err) {
    return console.error(err);
  }
console.log("Data overwrite using write method successfully!");
console.log("Let's read newly overwritten data of same file");
fs.readFile('test3.txt', function (err, data) {
    if (err) {
      return console.error(err);
    }
console.log("Asynchronous read: " + data.toString());
  });
});
```

Output:


```
C:\Users\User\node>node write2update_demo.js
Trying to overwrite an existing file with writeFile() method of File System
Data overwrite using write method successfully!
Let's read newly overwritten data of same file
Asynchronous read: New Overwritten text!
```

5.5.4 Delete Files

- The File System module has unlink() method for deleting files.
- **Syntax:**

  fs.unlink(path, callback)

  - path – string having the filename includ ng path
  - callback – This is the callback function which gets a single parameter err that returns an error in case of any deleting error

- **Example:**
- The following code deletes the file "text3.txt".Save this file as delete_demo.js.

```
var fs = require("fs");

console.log("Trying to delete an existing file");
fs.unlink('test3.txt', function(err) {
  if (err) {
    return console.error(err);
  }
console.log("File deleted successfully!");
});
```

Output:

```
C:\Users\User\node>node delete_demo.js
Trying to delete an existing file
File deleted successfully!
```

5.5.5 Rename Files

- The **fs.rename() method** is used to asynchronously rename a file at the given old path to a given new path. It will overwrite the destination file if it already exists.
- **Syntax:**

  fs.rename(oldPath, newPath, callback)

  - oldpath – path of the file that has to be renamed. It can be a string, buffer or URL
  - newPath - new path that the file has to be renamed. It can be a string, buffer or URL.
  - callback – This is the callback function which gets a single parameter err that returns an error in case of any renaming
- **Example:**
- The following code renames the file "text3.txt" to "temp.txt".Save this file as rename_demo.js.

```
var fs = require('fs');


console.log("Trying to rename an existing file");
fs.rename('test3.txt', 'temp.txt', function (err) {
if (err) {
  return console.error(err);
}
console.log("File renamed successfully!");
});
```

Output:

```
C:\Users\User\node>node rename_demo.js
Trying to rename an existing file
File renamed successfully!
```


5.6 Exercise

MCQ

A) Choose the correct option from following MCQs:

1. In which of the following areas, Node.js is absolutely perfect to use?
   A. I/O bound Applications
   B. Data Streaming Applications
   C. JSONAPI based Applications
   D. All of the above.

   Ans : D

2. Which of the following statement is valid to use a Node module fs in a Node based application?
   A. var fs = require("fs");
   B. var fs = import("fs");
   C. package fs;
   D. import fs;

   Ans: A

3. Which of the following statement is correct for Node js?
   A. It is Client Side Language.
   B. It is the Server Side Language.
   C. It is both Server Side and Client Side Language.
   D. None of the above.

   Ans :B

4. Node JS is written in _____ language
   A. C++
   B. C
   C. JavaScript
   D. All of the above

   Ans :D

5. Which of the following is the core module of Node JS
   A. MongoDB
   B. Express
   C. NPM
   D. package .json

   Ans : C

6. What is Callback?
   A. The callback is a technique in which a method calls back the caller method.
   B. The callback is an asynchronous equivalent for a function.

C. Both of the above.
D. None of the above.

Ans :B

7. Which of the following extension is used to save the Node.js files?
   A. .js
   B. .node
   C. .java
   D. .txt

   Ans: A

8. Which of the following is not the build-in module?
   A. http
   B. fs
   C. path
   D. fsread

   Ans :D

9. What does fs module stand for?
   A. File Service
   B. File Storing
   C. File System
   D. File Sharing

   Ans : C

10. Which of the following method of fs module is used to create file?
    A. fs.writeFile()
    B. fs.open()
    C. fs.appendFile()
    D. All of the above

    Ans :D

11. Node js is _____.
    A. Asynchronous
    B. Light Weight
    C. Highly Scalable
    D. All of the above

    Ans :D

12. Node js is multithreaded.
    A. False
    B. True

    Ans: A

13. Which function is used to include modules in node js?

A. require()
B. include()
C. import()
D. attach()

Ans: A

14. How node js modules can be created externally?
    A. module.expose
    B. module.exports
    C. module.spread
    D. None of the above

    Ans:B

15. What does NPM stands for?
    A. Node Package Manager
    B. Node Project Manager
    C. New Package Manager
    D. New Project Manager

    Ans: A

16. Which command shows the version of npm
    A. $ npm – version
    B. $ node –version
    C. $ npmgetversion
    D. $ node getverion

    Ans: A

B) Answer in brief:

1. Define npm
2. What is package .json?
3. Which commands are used to check version of node and npm?
4. Where Node Js is used for?
5. Give examples of few third party modules.
6. Name few tools and frameworks developed over Node JS to make it more easier to use.
7. What is the use of require()?
8. Explain writeHead() and createServer() methods of HTTP modules.

C) Answer in detail:

1. Explain important concept, features and working of Node JS.

2. What is working of http module explain in detail.
3. Discuss various components of Node JS.
4. List and explain few methods of request and response objects of http.
5. Explain how to create and include user-defined modules.
6. Explain Node Js as a server.
7. Give a detailed description of different ways to create a file.

****