Table of Contents

CHAPTER 1
Fundamentals of Web Technology

Some Old Questions:

a) Is the Internet and web same? Write about the difference between internet, intranet and extranet.

b) Differentiate between HTML and XHTML.

c) What happens if you enter https://pu.edu.np into your browser? Describe the procedures behind the screens. Describe along with its block diagram.

d) Discuss the terminologies HTTP,POP and SMTP in web technology in detail.

e) Define DNS. List the different MIME headers and explain them.

f) Discuss the need of web browser and we server in accessing the Internet.

g) What protocol is used by all computer connections to the Internet? Describe a fully qualified domain name with example.

h) What is the purpose of MIME type specification in a request/response transaction between a browser and a server?

i) Explain in brief about the domain name registration and hosting process.

j) How does HTTP works ? Describe.

k) Define DNS. Explain about HTTP and MIME protocols.

l) A company asked you to develop website for it. Which language and tools will you use for the development and why? Also how can you put the website online?

m) What is the function of DNS? **Write about the differences between LAN, WAN, MAN, PAN and web.**

n) Explain the following:
   i. URL vs URI
   ii. MIME Headers
   iii. HTTP vs HTTPs
   iv. SMTP vs POP3

o) Short Note On:
   i. Domain Registration Process
   ii. MIME
   iii. SMTP
   iv. Web Hosting
   v. RSS Feed
   vi. FTP

What is web technology?

- We probably know that computers don't communicate with each other the way that people do.
- That mean computers require codes, or directions. These binary codes and commands allow computers to process needed information.
- So to establish communication between computers Web Technology was invented.
- **Web Technology refers the methods that communicate computers with each other through the use of markup language and multimedia packages.**

Markup Language:

- **A markup language is a computer language that uses tags to define elements within document.**
- It is human-readable and markup files contain standard words , rather than typical programming syntax.
- Among several markup languages HTML and XML are most popular.
- Html is used for creating webpages.
- XML is used for storing structured data , rather than formatting information on a page.

Multimedia Packages:

- **Multimedia is content that uses a combination of different content forms such as text, audio, images, animations, video and interactive content.**
- Multimedia contrasts with media that use only rudimentary computer displays such as text-only or traditional forms of printed or hand-produced material.
- Multimedia devices are electronic media devices used to store and experience multimedia content.

Internet:

- The concept of a new large-scale computer network was first developed by the U.S Department of Defense (DoD), in the 1960s.
- The main purpose of this network were communications , program sharing , and remote computer access for researchers working on defense-related contracts.
- NSFNET was decommissioned in 1995, removing the last restrictions on the use of the Internet to carry commercial traffic.
- **That means, internet is a globally connected network system that uses TCP/IP to transmit data via various types of media.**
- The internet is a network of global exchanges – including private, public, business, academic and government networks – connected by guided, wireless and fiber-optic technologies.
- The terms **internet** and **World Wide Web** are often used interchangeably, but they are not exactly the same thing; the internet refers to the global communication system, including hardware and infrastructure, while the web is one of the services communicated over the internet.

- Internet Protocol (IP) is the internet’s primary component and communications backbone.
- IP is used to allocate the devices that are connected to the network.

WWW(World Wide Web):

- The World Wide Web ("WWW" or simply the "Web") is a global information medium which users can read and write via computers connected to the Internet.
- The **World Wide Web** (**WWW**), is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and accessible via the Internet.
- An information space is a type of information design in which representations of information objects are situated in a principled space.
- A new protocol for the Internet , as well as a system of document access to use it was proposed by a small group of people led by Tim Berners-Lee at CERN(Conseil Europeen pour la Recherche Nucleaire) , In 1989.
- The World Wide Web has been central to the development of the Information Age and is the primary tool billions of people use to interact on the Internet.
- Hypertext is the text that is specially coded using a standard coding language known as HyperText Markup Language.

Web Browsers:

- In some network , when two computers communicate , in many cases one acts as client and the other as a server.
- First the communication initiates by client , which is often a request for information stored on the server which sends information back to the clients.
- Browsers are programs running on client machines where documents are requested to server.
- A **web browser** (commonly referred to as a **browser**) is a software application for accessing information on the World Wide Web.
- They are called browsers because they allow the user to browse the resources available on the servers.
- The first browsers were text based – they were not capable of displaying graphic information nor did they have a graphical user interface.

Web Servers:

- Web servers are programs that provide documents to requesting browsers identified using URLs.
- Servers are slave Programs : They act only when requests are made to them by browser running on others computers on the Internet.
- The most commonly used Web servers are Apache , which runs under Windows operating systems.

1. **Web Server Operation:**
   - First, Web browsers initiate network communications with servers by sending URLs.
   - One of two different things can be specified by URL: the address of data file stored on the server or a stored program on the server that clients wants executed and the output of the program returned to the client.
   - The standard Web protocol HTTP(Hypertext Transfer Protocol), is responsible for all the communication between Web client and Web servers.
   - When Web server begins execution , it informs the operating system under which it is running that it is now ready to accept incoming network connections through a specific port on the machine.
   - In this running state , the server runs as a background process in the operating system.
   - Then a network connection is established between Web client or browser and Web server.
   - And sends information requests and possibly data to the server , receives information from the server and close the connection.

2. **General Server Characteristics:**
   - Document root
     The file structure of the serer has two separate directories. The root of one of this is document root.
   - Server root
     This is another root directory of web server file structure.
   - Virtual document tree
     Many servers allow a part of the servable document stored out side of the document root this is called virtual document tree.
   - Virtual hosts
     Many servers can support more than one site on a computer potentially reducing the cost of each site and makes their maintenance more convenient. such secondary hosts are called virtual hosts.
   - Proxy server
     Some servers can serve documents that are in the document root of the other machines are on the web are called proxy servers.
     Web servers not only support HTTP but also ftp, Gopher, News and mail.

URL(Uniform Resource Locators)

- Uniform (or Universal) Resource Locators (URLs) are used to identify document (resources) on the Internet.

- Due to the URL address, the user gets information about where the needed information is located.

- It can refer to the website, some particular document, or an image.

- It is also referred to as a web address.

- URLs consist of multiple parts -- including a protocol and domain name -- that tell a web browser how and where to retrieve a resource.

- End users use URLs by typing them directly into the address bar of a browser or by clicking a hyperlink found on a webpage, bookmark list, in an email or from another application.
- URIs are strings of characters used to identify a resource over a network.

1. URL structure

- The URL contains the name of the protocol needed to access a resource, as well as a resource name.
- The first part of a URL identifies what protocol to use as the primary access medium.
- The second part identifies the IP address or domain name -- and possibly subdomain -- where the resource is located.

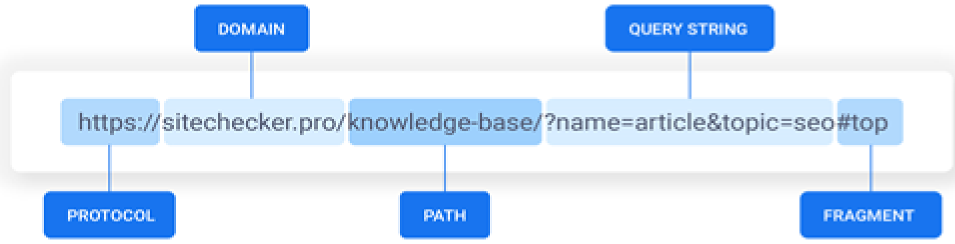

DOMAIN
QUERY STRING
https://sitechecker.pro/knowledge-base/?name=article&topic=seo#top
PROTOCOL
PATH
FRAGMENT

figure: Example of URLs

Optionally, after the domain, a URL can also specify:

- a path to a specific page or file within a domain;
- a network port to use to make the connection;
- a specific reference point within a file, such as a named anchor in an HTML file; and
- a query or search parameters used -- commonly found in URLs for search results.

MIME(Multipurpose Internet Mail Extensions)

- A browser needs some way of determining the formats of document it receives from a Web servers.
- It is not possible to render documents without knowing the formats of the document because different document formats require different rendering software.
- **MIME** was developed to specify the format of different kinds of documents to be sent via Internet mail.
- **MIME (Multi-Purpose Internet Mail Extensions)** is an extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds

of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol.

- A MIME type consists of a **type** and a **subtype** — two strings separated by /.
- The *type* represents the category and can be a *discrete* or a *multipart* type. The *subtype* is specific to each type.

Discrete types indicate the category of the document. They can be one of the following table:

Type	Description	Example of typical subtypes
text	Any document that contains text and is theoretically human readable	text/plain, text/html, text/markdown
image	Any kind of image. Videos are not included, though animated images (like animated GIF) are described with an image type.	image/gif, image/png, image/jpeg, image/bmp, image/vnd.microsoft.icon
audio	Any kind of audio file	audio/midi, audio/mpeg, audio/webm, audio/ogg, audio/wav
video	Any kind of video file	video/ogg, video/webm
application	Any kind of binary data, especially data that will be executed or interpreted somehow.	application/javascript, application/octet-stream, application/pkcs12, application/vnd.mspowerpoint, application/xml,application/pdf, application/xml, application/xhtml+xml

HTTP(Hypertext Transfer Protocol)

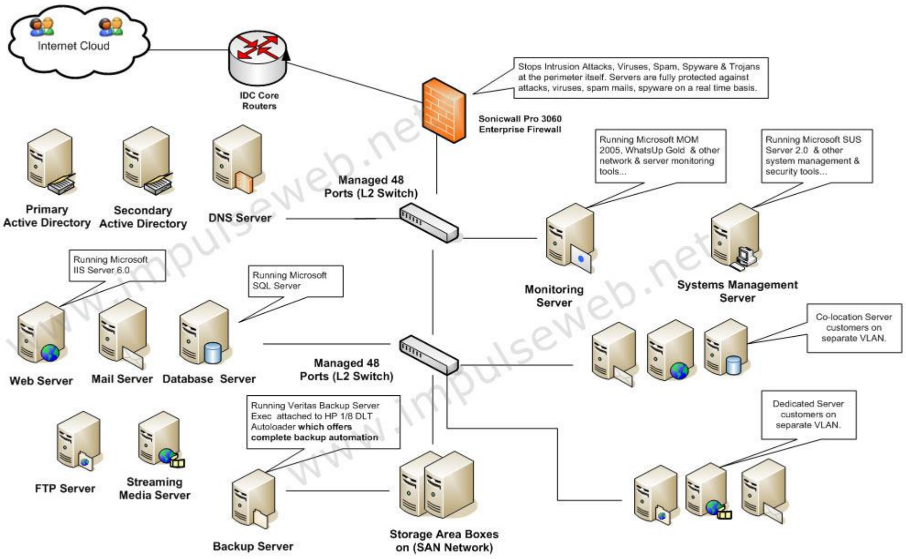
- HTTP is the protocol that is used for transferring hypertext (i.e text ,graphic , audio, video ,animation etc) between two computers and is particularly used on the World Wide Web.
- It is a TCP/IP based communication protocol and provides a standard for Web browsers and servers to communicate.
- Where as Hypertext is the text that is specially coded using a standard coding language called Hypertext Markup Language(HTML) which basically creates hyperlinks and thereby controls how the World Wide Web works and how Webpages are formatted and displayed.
- These hyperlinks can be in the form of text, graphic , image ,sound or video and are used to “link” the user to some other file.
- HTTP defines how message formatted and transmitted and what actions Web servers and browsers should take in response to various commands .
- For example , when you enter a URL in your browser , this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Webpage.
- HTTP is based on Client/Server principle .
- Communication between the host and the client occurs through a request/response pair.

- A connection is established between two computers- out of which one is client (generally the browser) that initiates the request and the other is the server that responds to the request.
- Also, HTTP identifies the resource that the client has requested for and informs the server about the action to be taken.
- When the user clicks on the hypertext link ,the client program on their computer uses HTTP to contact the server , identify the resource and ask the server to respond with an action.

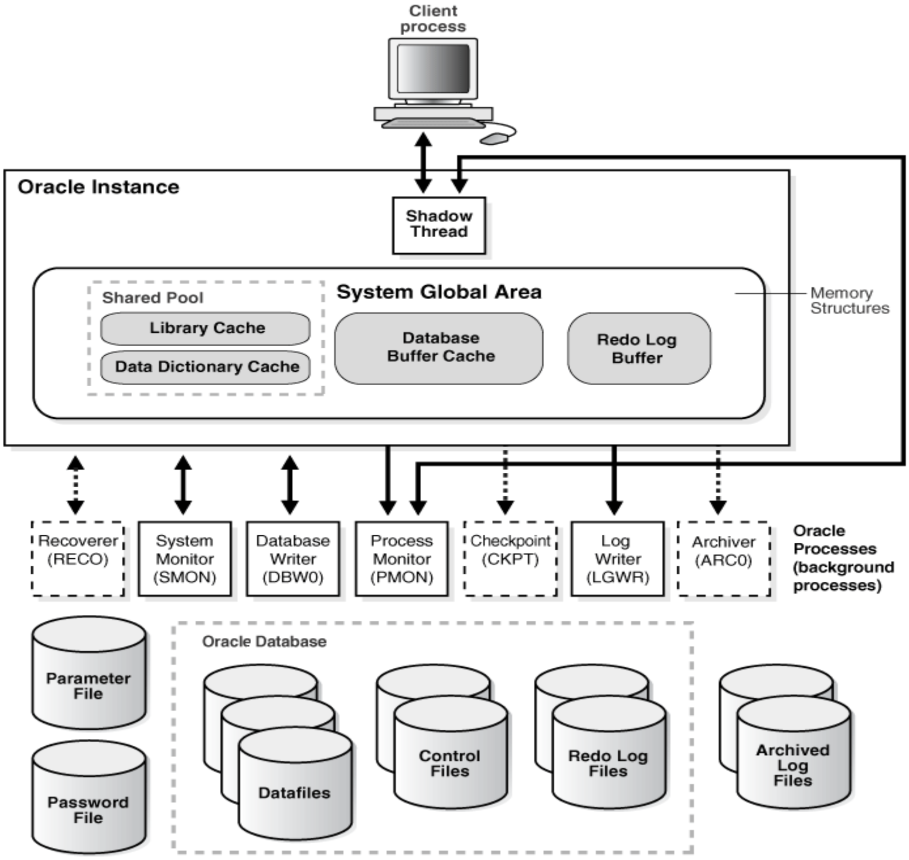
  HTTP has three important features.
- **Firstly,** it is connectionless .After a request is made , the client disconnects from the server and waits for a response. To process the request , the server has to re-establish the connection with the client .
- **Secondly**, HTTP is media independent. This means any type of data(text , audio, video, etc) can be sent by HTTP as long as both the client and server know how to handle the data content.
- **Thirdly,** HTTP is stateless. This is because the server and the client are aware of each other only during request. Afterwards , they get disconnected.

Web Architecture

- Before we can get to define what **web architecture** is, first it is necessary to frame it properly.
- There are some roles related to architecture within the Information Technology industry, but basically we can do the following division:
  - **<u>System architects:</u>** combine different hardware components (computers and other devices) with software elements (operating systems) to build systems capable of providing the resources needed by applications or services which run on top of them.


Internet Cloud
IDC Core Routers
Stops Intrusion Attacks, Viruses, Spam, Spyware & Trojans at the perimeter itself. Servers are fully protected against attacks, viruses, spam mails, spyware on a real time basis.
Sonicwall Pro 3060 Enterprise Firewall
Running Microsoft MOM 2005, WhatsUp Gold & other network & server monitoring tools...
Running Microsoft SUS Server 2.0 & other system management & security tools...
Primary Active Directory
Secondary Active Directory
DNS Server
Managed 48 Ports (L2 Switch)
Monitoring Server
Systems Management Server
Running Microsoft IIS Server 6.0
Running Microsoft SQL Server
Co-location Server customers on separate VLAN.
Web Server
Mail Server
Database Server
Managed 48 Ports (L2 Switch)
Running Veritas Backup Server Exec attached to HP 1/8 DLT Autoloader which offers complete backup automation
Dedicated Server customers on separate VLAN.
FTP Server
Streaming Media Server
Backup Server
Storage Area Boxes on (SAN Network)

- **Data architects:** design how information handled by applications is structured using databases.



- **Storage Architects:** design storage area networks (SAN) that allow to store all the information generated by applications running on different information systems.
- **Networking Architects:** plan and design communication networks that allow data exchange between different information systems.
- **Software Architects:** more typically known as **software engineers**, these professionals design and build applications that provide services needed by users of information systems or other applications.

Web architecture determines how software logic is divided between a server and a client, as well as how these two components communicate with each other. The ways web architecture types work are quite different and depend on a client model and requests a client sends/receives to/from a server.

Depending on the needs of your web site or a web app, you choose the web architecture that would work best for it. Some most conceived part of web architect are:

- **Designing the user interface** of web applications. In the case of a website, it would be the design of the website itself: look & feel (colors, images, fonts used, etc.), page layout, menu structure, visual elements, etc.

- **Design and implementation of application logic**, i.e. the set of capabilities which it provides, such as user-entered **data processing**, **results calculation** from different input data, **algorithms designing and implementation**, **manipulation of information** stored in databases, **executing different actions** as a result of compliance with some given conditions or event triggering, etc. i.e. planning and designing of what later is implemented using one or more **programming languages**.
- **Designing of information architecture**, i.e. determine the real-world information that the application will be able to handle, design a **conceptual model** which is an accurate reflection of the real world with its different **entities and relationships** between those entities, determine the **data model** which best suits that conceptual model, implement the data model on a particular **database engine** and move the necessary information to it for the application proper functioning.

WAP(Wireless Application Protocol)

- WAP is a technical standard for accessing information over a mobile wireless network.
- A WAP browser is a web browser for mobile devices such as mobile phones that uses the protocol.
- WAP (Wireless Application Protocol) is a specification for a set of communication protocols to standardize the way that wireless devices, such as cellular telephones and radio transceivers, can be used for Internet access, including e-mail, the World Wide Web, newsgroups, and instant messaging.
- Instant messaging , is the exchange of near real-time messages through a stand-alone application or embedded software.
- WAP only access text rather than images or video or audio etc, that means WAP can provides us graphic interface.
  The WAP layers are:
  - Wireless Application Environment (WAE)
  - Wireless Session Layer (WSL)
  - Wireless Transport Layer Security (WTLS)
  - Wireless Transport Layer (WTP)

The WAP was conceived by four companies: Ericsson, Motorola, Nokia, and Unwired Planet (now Phone.com). The Wireless Markup Language (WML) is used to create pages that can be delivered using WAP.

Web standards

- Web standards are the formal , non-proprietary standards and other technical specifications that define and describe aspects of the WWW.

- Web standards are rules and guidelines established by the World Wide Web Consortium(W3C) developed to promote consistency in the design code which makes up a web page.

  The advantages in adhering to these standards are many:

  - Web pages will display in a wide variety of browsers and computers, including new technology.
  - W3C standards promote the use of "Cascading Style Sheets"(CSS) or design code which is attached to the web page. The use of style sheets significantly reduces the page file size.
  - Design features such as colors and fonts can be easily changed by just modifying one style sheet instead of editing every individual page in a website, reducing the costs to modify site.
  - Search engines are able to access and index pages designed to web standards with greater efficiency.

Email Protocols

SMTP(Simple Mail Transfer Protocol):

- Simple Mail Transfer Protocol (SMTP) is the standard protocol for **sending emails** across the Internet.
- Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another; the messages can then be retrieved with an e-mail client using either POP or IMAP.
- Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another; the messages can then be retrieved with an e-mail client using either POP or IMAP.

By default, the SMTP protocol works on three ports:

- **Port 25** - this is the default SMTP non-encrypted port
- **Port 2525** - this port is opened on all Site Ground servers in case port 25 is filtered (by your ISP for example) and you want to send non-encrypted emails with SMTP
- **Port 465** - this is the port used if you want to send messages using SMTP securely

POP(Post Office Protocol)

- POP is a protocol that is used to retrieve mail from mail server.
- POP access mails that are only located in inbox.
- Post Office Protocol version 3 (POP3) is a standard mail protocol used to **receive emails** from a remote server to a local email client.
- POP3 allows you to download email messages on your local computer and read them even when you are offline and removed from the email server.
- This means if you want to access your mail from multiple location , you can't.
- That means POP do not allow multiple user.

By default, the POP3 protocol works on two ports:

- **Port 110** - this is the default POP3 non-encrypted port
- **Port 995** - this is the port you need to use if you want to connect using POP3 securely

IMAP(Internet Message Access Protocol)

- The **Internet Message Access Protocol** (**IMAP**) is an Internet standard protocol used by email clients to retrieve email messages from a mail server over a TCP/IP connection.
- Stores email messages on a mail server, but allows the end user to view and manipulate the messages as though they were stored locally on the end user's computing device(s).
- This allows users to organize messages into folders, have multiple client applications know which messages have been read, flag messages for urgency or follow-up and save draft messages on the server.
- Most implementations of IMAP support multiple logins; this allows the end user to simultaneously connect to the email server with different devices.

By default, the IMAP protocol works on two ports:

- **Port 143** - this is the default IMAP non-encrypted port
- **Port 993** - this is the port you need to use if you want to connect using IMAP securely

FTP (File Transfer Protocol)

- The **File Transfer Protocol** (**FTP**) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.
- FTP is built on a client-server model architecture using separate control and data connections between the client and the server.
- FTP uses the Internet's TCP/IP protocols to enable data transfer.
- FTP promotes sharing of files via remote computers with reliable and efficient data transfer.
- Files can be transferred between two computers using FTP software.
- The user's computer is called the local host machine and is connected to the Internet.

- The second machine, called the remote host, is also running FTP software and connected to the Internet.
- A user typically needs to log on to the FTP server, although some servers make some or all of their content available without login, also known as anonymous FTP.
- FTP is less secure protocol because it uses plain text format when transferring files.

Web Hosting

- Web hosting is a service that allows organizations and individuals to post a website or web page onto the Internet.
- Web hosts are companies that provide space on a server owned or leased for use by clients, as well as providing Internet connectivity, typically in a data center.
- Websites are hosted, or stored, on special computers called servers.

Types of hosting

- **Shared web hosting service** refers to a web hosting service where many websites reside on one web server connected to the Internet.
- **Reseller hosting** is a form of web hosting wherein the account owner has the ability to use his or her allotted hard drive space and bandwidth to host websites on behalf of third parties.
- A **virtual private server** (**VPS**) is a virtual machine sold as a service by an Internet hosting service.
- A **dedicated hosting service**, **dedicated server**, or **managed hosting service** is a type of Internet hosting in which the client leases an entire server not shared with anyone else.
- A **colocation centre** (also spelled **co-location**, or **colo**) or "**carrier hotel**", is a type of data centre where equipment, space, and bandwidth are available for rental to retail customers.
- **Cloud computing** is shared pools of configurable computer system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet.

Assignment:

1. Write short note about HTTP.
2. Explain Domain name and its hierarchy.
3. Write about domain name registration process.
4. Intranet and Extranet.
5. Write about the differences between LAN, WAN, MAN, PAN and web.

Additional Topics:

Free source software:

- The Free Software Definition written by Richard Stallman and published by Free Software Foundation (FSF), defines free software as being software that ensures that the end users have freedom in using, studying, sharing and modifying that software.
- Free and open-source software (FOSS) is software that can be classified as both free software and open-source software.
- That is, anyone is freely licensed to use, copy, study, and change the software in any way, and the source code is openly shared so that people are encouraged to voluntarily improve the design of the software.

Unlike the Open Source term, Free Software only has 4 "Freedoms" with its definition and are numbered 0-3:

1. The freedom to run the program for any purpose (Freedom 0)
2. The freedom to study how the program works and adapt it to your needs (Freedom 1)
3. The freedom of redistribution of software (Freedom 2)
4. The freedom to improve the program and release your improvements to the public to benefit the while community. (Freedom 3)

Open source software:

- Open-source software (OSS) is computer software distributed with its source code available for modification.
- The software usually includes a license for programmers to change the software in any way they choose.
- They can fix bugs, improve functions, or adapt the software to suit their own needs.
- The Open Source Initiative (OSI) is a leading authority on OSS
- For software to be considered "Open Source" it must meet ten conditions. Among these ten conditions the first three that are really at the core of Open Source and differentiates it from other software.
  - Free Redistribution: the software can be freely given away or sold.
  - Source Code: the source code must either be included or freely obtainable.
  - Derived Works: redistribution of modifications must be allowed.

OSI: Open Source Initiative (OSI) is a non-profit organization dedicated to promoting open-source software.

Advantages of Open-Source Software

While cost is a driving factor, OSS has several additional benefits:

- High-quality results when the source code is passed around, tested and fixed.

- It is a valuable learning opportunity for programmers. They can learn and apply skills to the most popular programs available today.
- Many consider open-source software more secure than proprietary software because bugs are identified and fixed quickly.

Most of the software is free. Costs may arise later, however, such as subscriptions or support fees.

Popular Types of Open-Source Software

- Mozilla's Firefox web browser
- Thunderbird email client
- PHP scripting language
- Python programming language
- Apache HTTP web server
- database system

Proprietary Software:

- Proprietary software is software that is owned by an individual or a company (usually the one that developed it).
- There are almost always major restrictions on its use, and its source code is almost always kept secret.
- Source code is the form in which a program is originally written by a human using a programming language and prior to being converted to machine code which is directly readable by a computer's CPU (central processing unit).
- It is necessary to have the source code in order to be able to modify or improve a program.
- Virtually all Microsoft software is proprietary, including the Windows family of operating systems and Microsoft Office.
- This includes software that is given away at no charge, such as Internet Explorer.
- Other major producers of proprietary software include Adobe, Borland, IBM, Macromedia, Sun Microsystems and Oracle.

Advantages:

- One advantage to using a proprietary-software system is that you will generally be able to take advantage of the software company's customer service department for troubleshooting and setup purposes.

Disadvantage:

- These types of software are costly and not always needed.

Chapter 2

Introduction to HTML and XHTML

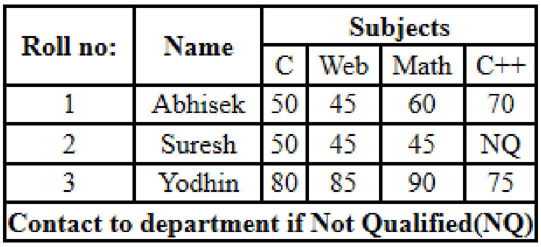
Some Old Questions:

a) Write basic structure of HTML. How HTML differs from XHTML?

b) What is image map in HTML? Write a HTML code to show the mapping of image.

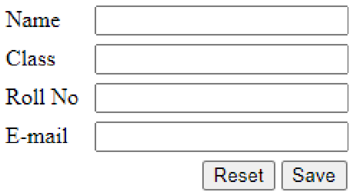
c) Create an HTML form with one textbox for getting name, one textbox for getting password, two radio buttons for getting gender(male and female),dropdown for getting subjects(Web, Math) and *textarea* for getting feedback.

d) Why table-less design is preferred more than table-based design ? Write HTML code for following :



Roll no:	Name	Subjects			
		C	Web	Math	C++
1	Abhisek	50	45	60	70
2	Suresh	50	45	45	NQ
3	Yodhin	80	85	90	75
Contact to department if Not Qualified(NQ)					

e) Write HTML tags to generate the following form:

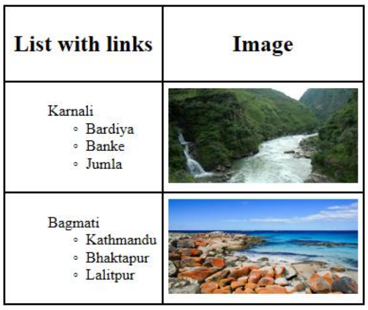


Name	
Class	
Roll No	
E-mail	

f) Create an HTML document that contains five different headlines from a newspaper headline. Each headline should use a different heading tag and a different color. Make all the text use the Helvetica font. Every verb in the text must have a line through it.

g) What is Frame? List out the advantages and disadvantages of using a Frame.

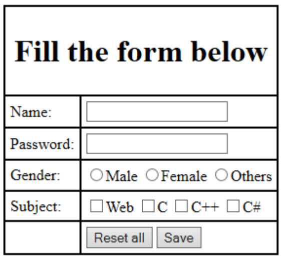
h) Write a HTML code for following:



List with links	Image
Karnali ◦ Bardiya ◦ Banke ◦ Jumla	
Bagmati ◦ Kathmandu ◦ Bhaktapur ◦ Lalitpur	

Also embedded CSS properties and property value as needed.

i) Write a HTML code to generate following table with form elements.



Fill the form below

Name:	
Password:	
Gender:	○Male ○Female ○Others
Subject:	□Web □C □C++ □C#
	Reset all Save

Also embedded CSS properties and property value as needed.

j) What is the difference in behavior between a group of checkbox buttons and a group of radio buttons? Under what circumstances select menu is used instead of a radio button group?

k) Short Notes On:

i) List in HTML
ii) Frameset
iii) Multimedia in HTML
iv) Textarea

HTML (Hypertext Markup Language):

- **Hypertext Markup Language** (**HTML**) is the standard markup language for creating web pages and web applications.
- With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.
- Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.
- HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
- HTML elements are the building blocks of HTML pages.
- HTML elements are represented by tags.
- Browsers do not display the HTML tags, but use them to render the content of the page.
- Hypertext Markup Language revision 5 (HTML5) is markup language for the structure and presentation of World Wide Web contents.
- HTML5 supports the traditional HTML and XHTML-style syntax and other new features in its markup, New APIs, XHTML and error handling.

Example: Basic syntax

```
<!DOCTYPE html>
<html>
    <head>
          <title>Basic structure of HTML</title>
    </head>
<body>
    <h1>First Heading</h1>
    <p>This tag is used for paragraph.</p>
</body>
</html>
```

- The <!DOCTYPE html>declaration defines this document to be HTML5
- The <html>element is the root element of an HTML page
- The <head>element contains meta information about the document

- The <title>element specifies a title for the document
- The <body>element contains the visible page content
- The <h1>element defines a large heading
- The <p>element defines a paragraph

HTML Tags:

- HTML tags are element names surrounded by angle brackets.
- HTML tags normally come **in pairs** like <p> and </p> .
- The first tag in a pair is the **start tag,** the second tag is the **end tag.**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name.

  <tagName> Content here……</tagName>
  The start tag is also called opening tag , and the end tag is also called closing tag.

XHTML

- **Extensible Hypertext Markup Language** (**XHTML**) is part of the family of XML markup languages.
- It mirrors or extends versions of the widely used Hypertext Markup Language (HTML), the language in which Web pages are formulated.
- XHTML is almost identical to HTML.
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers.

The Most Important Differences from HTML:

Document Structure

- XHTML DOCTYPE is **mandatory**
- The xmlns attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> are **mandatory**

XHTML Elements

- XHTML elements must be **properly nested**
- XHTML elements must always be **closed**
- XHTML elements must be in **lowercase**
- XHTML documents must have **one root element**

XHTML Attributes

- Attribute names must be in **lower case**
- Attribute values must be **quoted**
- Attribute minimization is **forbidden**

Standard HTML Document Structure

- The first line of every HTML document is a DOCTYPE command , which specifies the particular SGML document -type definition (DTD) with which document complies.

- SGML is a system for defining markup languages. The DTD defines the syntax of markup constructs
- An HTML document must include the four tags *<html>* , *<head>* , *<title>*, and *<body>* .
- The *<html>* tag identifies the root element of the document ,
- So, HTML documents always have an *<html>* tag following the DOCTYPE command and they always end with the closing html tag, *</html>*.
- The html element includes an attribute , lang , which specifies the language in which the document is written. (*<html lang="en"> means English language*)
- The *head* element is the first element to appear after the opening *html* tag.
- In the document *head* we place things like the page *title* and *meta* data, we add JavaScript to our page with the *script* tag, and we *link* to external stylesheets and other resources.
- All of the content that is visible on a web page is nested between opening and closing *body* tags. The body is the primary container of the content that makes up a web page.
- To display an HTML page correctly, a web browser must know the character set used in the page.
- The most popular international character set used for the web is the 8-bit Unicode Transformation Format (UTF-8) .
- This character set uses from one to six bytes to represent a character , but is backward compatible with the ASCII character set.
- The meta tags specifies the character set used to write the document. The following is the necessary meta element:
  (*< meta charset = " utf-8 " />*)

Basic Text Formatting

HTML also defines some special elements for defining text with in the document with special meaning.

- *<b>*- element defines **bold** text, without any extra importance.
- *<strong>*- element defines **strong** text, with added semantic "strong" importance.
- *<i>*- element defines *italic* text, without any extra importance.
- *<em>*- element defines *emphasized* text, with added semantic importance.
- *<mark>*- element defines marked or highlighted text.
- *<small>*- element defines smaller text .
- *<del>*- element defines ~~deleted~~ (removed) text.
- *<ins>*- element defines inserted (added) text.
- *<sub>*- element defines $_{\text{subscripted}}$ text.
- *<sup>*- element defines $^{\text{superscripted}}$ text.

Example:

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

```
        <b>This text is bold</b>
        <strong>This text is strong</strong>
        <i>This text is italic</i>
        <em>This text is emphasized</em>
        <h2>HTML <small>Small</small> Formatting</h2>
        <h2>HTML <mark>Marked</mark> Formatting</h2>
        <p>My favorite color is <del>blue</del> red.</p>
        <p>My favorite <ins>color</ins> is red.</p>
        <p>This is <sub>subscripted</sub> text.</p>
        <p>This is <sup>superscripted</sup> text.</p>

</body>
</html>
```

Basic Text Markup

- *Paragraphs* – Text is normally organized into paragraphs in the body of a document. < p> elements defines paragraph.
- *Line Breaks* -Sometimes we need text explicit line break without the preceding blank line. <br /> element defines line breaks. The slash indicates that the tag is both an opening and closing tag. The space before slash represents the absent content.
- Preserving White Space – Sometimes it is desirable to preserve the white space in text - .i.e. to prevent the browser from eliminating multiple spaces and ignoring embedded line breaks. <*pre* > tag is used </*pre*>
- *Headings* - There are six level of headings (*h1, h2, h3, h4, h5, & h6*) that are defined by HTML. Where <*h1*> specifies highest – level heading and <*h6*> specifies lowest heading. The heading tags always break the current line , so their content always appears on a new line.
- Block Quotations – Sometimes we want a block of text to be set off from the normal flow of text in a document . The HTML <*q*> element defines a short quotation . The HTML <*blockquote*> element defines a section that is quoted from another source.
- Horizontal Rules - Two parts of a document can be separated from each other by placing a horizontal line between then, such lines are called horizontal rules.
- Abbreviations - <*abbr*> element defines an abbreviation or an acronym.
- Address - <*address*> element defines contact information (author/owner) of a document or an article.

Example:

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

```
    For 50 years, WWF has been protecting the future of nature .The world's leading
    conservation organization, WWF works in 100 countries and is supported by 1.2 million
    members in the United States and close to 5 million globally.
</blockquote>
<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>
<address>
    Written by John Doe.<br>
    Visit us at:<br>
    Example.com<br>
    Box 564, Disneyland<br>
    USA
</address>
</body>
</html>
```

Images:

- The inclusion of images in a document can dramatically enhance its appearance , although images slow the document-download process.
- The file in which the image is stored is specified in a tag.
- There are the two most common methods of representing images are the Graphic Inter-change Format(GIF ,pronounced like the first syllable of jiffy) and the Joint Photographic Experts Group(JPEG) formats.
- Files in both formats are compressed to reduce storage needs and allow faster transfer over the Internet.
- Files containing GIF images use the *.gif* (or .GIF) extension on their names.
- The JPEG image file use *.jpg* (or .jpeg or JPG) extension on their names.
- The image element , whose tag is *<img />* , is an inline element that specifies an image that is to appear in a document .

<img src="image1.jpg" alt="Girl" style=" width:500px; height:600px; ">

Or

<img src="image1.jpg" alt="Girl" width:"500px" height:"600px ">

Where ***src*** *is an attribute that identifies the location/path (also called web address) of the source. The* ***alt*** *attribute provides an alternate text for an image, if the user for some reason cannot view it.*

Hypertext links

- A hypertext link in an HTML document , which we simply call a *link* here , acts as a pointer to some particular place in some Web resource.
- That resources can be HTML document anywhere on the Web , or it may be the document currently being displayed .
- Most of the web sites consist of many different documents , all logically linked together.
- A link that points to a different resource specifies the address of that resource.
- Links are specified in an attribute of an anchor element , ***a***, which is an inline element.
  <a href="https://www.google.com">Visit our HTML tutorial</a>
- Where (*Visit our HTML tutorial)* link text is the visible part on the browser. Clicking on the link text sending you to specified address.

- **Local links** - A local link (link to the same web site) is specified with a relative URL (without https://www....). Link file from your pc/server.

  <p><a href="html_images.asp">HTML Images</a> </p>
- **HTML Links - The target Attribute** - The target attribute specifies where to open the linked document.

  The target attribute can have one of the following values:

  - **_blank** - Opens the linked document in a new window or tab
  - **_self** - Opens the linked document in the same window/tab as it was clicked (this is default)
  - **_parent** - Opens the linked document in the parent frame
  - **_top** - Opens the linked document in the full body of the window
  - ***framename*** - Opens the linked document in a named frame

  <a href="https://www.google.com/" target="_blank">Visit!</a>
- **Image as Link -** Images are commonly used as link.

  <a href="https://wwww.facebook.com">

  <img src="image.jpg" alt="Image_Link" style="width:42px; height:42px; border:0"></a>
- **Link Titles** - The title attribute specifies extra information about an element. The information is most often shown as a tooltip text when the mouse moves over the element.

  <a href="https://www.gmail.com" title="Go to gmail">visit for mail to gmail</a>
- **HTML Link Colors -** By default, a link will appear like this (in all browsers):

  - An unvisited link is underlined and blue
  - A visited link is underlined and purple
  - An active link is underlined and red

  You can change the default colors, by using CSS:

```
<!DOCTYPE html>
<html>
<head>
        <style>
                a:link {
                        color: green;
                         background-color: transparent;
                        text-decoration: none;
                }
                a:visited {
                        color: pink;
                        background-color: transparent;
                         text-decoration: none;
                        }
                        a:hover {
                                    color: red;
                                    background-color: transparent;
                                    text-decoration: underline;
                        }
                        a:active {
                                    color: yellow;
                                    background-color: transparent;
                                    text-decoration: underline;
                        }
            </style>
            <title>Color links</title>
</head>
<body>
            <h2>Link Colors</h2>
            <p>You can change the default colors of links</p>
            <a href="html_image.html" target="_blank">HTML Images</a>
</body>
</html>
```

Lists

- There are mainly three sorts of html lists.(Unordered Lists, Ordered List and Definition Lists).

  1. **Unordered Lists:**
     - The <*ul*> tag , which is a block tag , creates an unordered list.
     - Each item in a list is specified with an <*li*> tag.
     - Any tag can appear in list item , including nested lists with in the <*ul*> tag.
     - Items are displayed , each list item is implicitly preceded by a bullet.
     - Also we can change type of Unordered list .(.i.e. *type=square or circle or disc or none*)

  2. **Ordered Lists:**
     - The <*ol*> tag , which is a block tag , creates an ordered list.
     - Each item in a list is specified with an <*li*> tag.
     - Any tag can appear in list item , including nested lists with in the <*ol*> tag.
     - Items are displayed , each list item is implicitly preceded by a numbers(1, 2, 3….).
     - Also we can change type of Unordered list .(.i.e. *type= i / a/A/I*)

  3. **Definition Lists:**

     - Definition lists are used to specify lists of terms and their definitions.
     - A definition list is given as the content of a *dl* element , which is a block element.
     - Each term to be defined in the definition list is given as the content of a *dt* element.
     - Definitions themselves are specified as the content of *dd* elements .

Example*: Unordered lists , Ordered lists and Definition lists.*

```
<!DOCTYPE html>

<html>

<head>

        <title>About Lists</title>

</head>

<body>

        <h2>Unordered List with Bullets(as default)</h2>

    <p>You can change type as needed. Unordered types (square , circle , disc and none)</p>

        <ul style="list-style-type:disc">

                <li>Coffee</li>

                <li>Tea</li>

                <li>Milk</li>

        </ul>


    <h2>Ordered List with Numbers(as default)</h2>

    <p>You can change type as needed.</p>

        <ol type="">

                <li>Coffee</li>

                <li>Tea</li>

                <li>Milk</li>

        </ol>

    <h3>Definition Lists</h3>

        <dl>

                <dt>122</dt>

                <dd>C Programming language</dd>

                <dt>122</dt>

                <dd>C Programming language</dd>

        </dl>

</body>

</html>
```

Tables

- Tables provide a highly effective way of presenting many kinds of information.
- A table is a matrix of cells.

- The cells in the top row often contain column labels , those in the leftmost column often contain row labels , and most of the rest of the cells contain the data of the table.
- The content of a cell can be almost any document element , including text, a heading , a horizontal rule , an image , or a nested table.
- An HTML table is defined with the <*table*> tag.
- Each table row is defined with the <*tr*>tag.
- A table header is defined with the <*th*> tag.
- By default, table headings are bold and centered.
- A table data/cell is defined with the <*td*> tag.

Example:

```
<!DOCTYPE html>
<html>
<head>
            <title>Creating table</title>
</head>
<body>
            <h2>Basic HTML Table</h2>
            <table style="width:100%">
                        <tr>
                                    <th rospan="2">Firstname</th>
                                    <th rospan="2">Lastname</th>
                                    <th rospan="2">Age</th>
                                    <th colspan="3">Subjects(Marks)</th>
                        </tr>
                        <tr>
                                    <th>C</th>
                                    <th>C++</th>
                                    <th>Web</th>
                        </tr>
                        <tr>
                                    <td>Ram</td>
                                    <td>Saud</td>
                                    <td>23</td>
                                    <td>50</td>
                                    <td>60</td>
                                    <td>70</td>
                        </tr>
                        <tr>
                                    <td>Shyam</td>
                                    <td>Chaudhary</td>
                                    <td>26</td>
                                    <td>89</td>
                                    <td>60</td>
                                    <td>70</td>
                        </tr>
                        <tr>
                                    <td>Bharat</td>
                                    <td>Rokaya</td>
                                    <td>28</td>
                                    <td>50</td>
                                    <td>90</td>
                                    <td>70</td>
                        </tr>
                </table>
</body>
</html>
```

Add following properties too:

```
<style>
table, th, td {
  border: 1px solid black;
 border-collapse: collapse;
}
th, td {
  padding: 15px;
}
th {
  text-align: left;
}
</style>
```

- Use the CSS border-spacing property to set the spacing between cells.
- Use the *colspan* attribute to make a cell span to many columns.
- Use the *rowspan* attribute to make a cell span to many rows.
- Use the *id* attribute to uniquely define one table.

Frames

Frames are used to define different sections where we can load a separate HTML document. With update of HTML version some elements also updated . Among which defining frame in browser differ with HTML version. i.e. <frameset> and <iframe> tag supported by different.

- Instead of using <*frame* > tag in HTML5 we uses <*iframe*> tag.

iframe

- Each <*frame*> in a <*frameset*> can have different attributes, such as border, scrolling, the ability to resize, etc.
- <*iframe* > tag is used to define inline frame.

Example:

```
<!DOCTYPE html>
<html>
        <head>  <title>How to define iframe</title>  </head>
<body>
        <h2>Iframe - Target for a Link</h2>
        <iframe height="300px" width="100%" src="frame.html" name="iframe_1"></iframe>
        <p><a href="https://www.google.com" target="iframe_1">Google </a> </p>
        <p>When the target of a link matches the name of an iframe, the link will open in the iframe.
        </p>
</body>
</html>
```

Frameset:

- Frames are used to divide browser window into different sections where we can load a separate HTML document.
- <*frameset*> tag was used in HTML 4 and is a collection of *frames.*
- The <*frame*> tag defines one particular window (frame) within a <*frameset*>.
- Each <*frame*> in a <*frameset*> can have different attributes, such as border, scrolling, the ability to resize, etc.

  Creating Frames: <*frameset* > tag is used to define frames instead of <*body*> tag. To define horizontal frames and vertical frames, *rows* and *cols* attributes are used. <*frame*> tag indicate frame and defines which document shall be open into the frame.

  Example 1: Here we define frame horizontally, <*frameset*> tag uses *rows* attribute . Also, for vertical frame use *cols* attribute. Value used in rows / cols can be in percentage(%), pixels(px),etc . Also , can use other attributes like *frameborder="1/0", border="5/10/…",framespacing="5/10/20/…".*

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML Frames </title>
</head>
   <frameset rows="20%,*,20%" frameborder="1" border="15">
   <!-- Three frame of size (20%,*,20%) = (20%,60%,20%) -->
           <frame src="./someJsExample/someJsExample.html"></frame>
           <frame src="./someJsExample/second_class/new.html"></frame>
           <frame src="./someJsExample/fifthClass/dragAndDrop.html"></frame>
   </frameset>
</html>
```

<frame> tag Attributes: *src="./abc.html", name="frame1", frameborder="0/1" ,marginwidth="30/40/3/…", noresize ="noresize", marginheight="2/3/80/30/…" , scrolling="yes/no" etc.*

Example 2: For vertical frames.

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML Frames </title>
</head>
   <frameset cols="20%,30,30%" frameborder="1" border="1" framespacing="40" >
           <!-- Three frames of size (20%,*,20%) = (20%,60%,20%) -->
           <frame src="./someJsExample/someJsExample.html" ></frame>
           <frame src="./someJsExample/second_class/new.html"></frame>
           <frame src="./someJsExample/fifthClass/dragAndDrop.html"></frame>
           <noframes>
                   <body>Your browser does not support.</body>
           </noframes>
   </frameset>

</html>
```

Note: *<noframes> tag content would be displayed if browser doesn't support frames.*

Forms

- Form is used to communicate information from a web browser to the server.
- Also form is a window or screen i.e. used to gathering information(data) and also visualized with different , numerous field or spaces to enter data.
- Generally , form is used to enter data.
- In HTML5 form is defined within the <*form*> tag and having different control attributes.

Example:

```
<html>
        <head> <title>Designing Form</title> </head>
<body>
        <h2>HTML Forms</h2>
        <form action="firstPage.php" target="_blank" method="post">
        Country name:<br>
        <input type="text" name="countryName" value="Nepal">
        <br>
        Capital name:<br>
         <input type="text" name="capitalName" value="Kathmandu">
         <br><br>
        <input type="submit" value="Submit">
        </form>
        <p>If you click the "Submit" button, the form-data will be sent to a page called
        "firstPage.php".</p>
</body>
</html>
```

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

Note: *The page that response when button is clicked is* firstPage.php *i.e. given below.*

```
         <!-- This is firstPage.php file -->
<!DOCTYPE html>
<html>
<head>
        <title>Response on clicking submit button</title>
</head>
<body>
        Country Name:<?php echo $_POST["countryName"]; ?><br>
        <hr>            <!-- define horizontal ruler -->
        Capital Name:<?php echo $_POST["capitalName"]; ?>
</body>
</html>
```

Attributes that are used to control form:

- *action* – is used to define action to be performed when the form is submitted.
- *method* – is used to specifies HTTP method(post and get) when submitting the form.

- *name* – is used to sent data to all (where needed). Also used to identify the form.
- *target* – is used to specifies the target address.

Some elements that are used within the form:

- *input* element – is used to take input from user on browser i.e. it provides input field. Input fields are defined by an attribute called *type(text, checkbox, radio, password, email, reset, submit, image, button , color ,file etc).*
- *fieldset* – is used to group related data in a form.
- *legend* – is used to define caption for <*fieldset*> element.
- *Textarea* – is used to define multi-line text input control. Sometime used in a form to collect user inputs like comments or reviews.

Example*: Radio button – let user to choose / select one of a limited choice*

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

Example*: Input type password – defines password field*

```
<form action="">
 User name:<br>
 <input type="text" name="userName">
 <br>
 User password:<br>
 <input type="password" name="psw">
 </form>
```

Example*: Input type reset – defines reset button that is used to reset form and set all form values to default.*

```
<form action="firstPage.php">
        Country Name:<br>
        <input type="text" name="countryName" >
        <br>
        Capital Name:<br>
        <input type="text" name="capitalName" >
        <br><br>
        <input type="submit" value="Submit">
        <input type="reset">
</form>
```

Example*: Input type checkbox – defines checkbox let a user select ZERO or MORE options of a limited number of choices.*

```
<form action="secondPage.php" method=”POST”>
        <input type="checkbox" name="nation[]" value="Nepal">My county name is
        Nepal.
        <br>
        <input type="checkbox" name=" nation[] " value="Kathmandu"      >      Our
capital is Kathmandu.
        <br><br>
        <input type="submit">
</form>
```

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

Note: *The page that response when button is clicked is secondPage.php i.e. given below.*

```
<?php
if (isset($_POST["submit"])) {
        if (!empty($_POST["nation "])) {
                echo '<h3> you have selected following vehicle </h3>';
                foreach ($_POST["nation "] as $ nation) {
                        echo '<p>'.$ nation.'</p>';
                }
        }
        else {
                echo "<br>Please select at least one";
        }
}
?>
```

Example: ***<textarea>*** *tag*

```
<!DOCTYPE html>
<html>
<head>
        <title>TextArea Tag.</title>
</head>
<body >
        <form>
                <label>FullName:</label></br>
                <input type="text" name="fullName"></br>
                <label>Email:</label></br>
                <input type="text" name="email"></br>
                <label>Comments:</label></br>
                <textarea rows="3" cols="22" placeholder="Leave your comments here.....">

                </textarea>
        </form>
</body>
</html>
```

Multimedia in HTML

- Multimedia is content that uses the combination of different contents form such as audio , video , text, animation , graphic etc .
- Also different contents form have different formats.
- Almost anything that can be heard or seen called multimedia.
- Different types and formats of multimedia forms are often used by web page.

The Video Element

- There are 3 supported video format in HTML 5: MP4 , WebM , and Ogg.
- The browser support for the different formats is:

Browser	MP4	WebM	Ogg
Internet Explorer	Yes	No	No
Chrome	Yes	Yes	Yes
FireFox	Yes	Yes	Yes

Safari	Yes	No	No
Opera	Yes	Yes	Yes

- The HTML5 <video>element specifies a standard way to embed a video in a web page.
  - HTML video-media types

File Format	Media Types
Mp4	video/mp4
Ogg	video /ogg
WebM	video /webm

Example:

```
        <!DOCTYPE html>
<html>
<body>
<video width="320" height="240" controls>
        <source src="movie.mp4" type="video/mp4">
        <source src="movie.ogg" type="video/ogg">
        Your browser does not support the video tag.
</video>
</body>
</html>
```

How it works:

- The *controls* attribute adds video controls, like play, pause, and volume.
- It is good idea to include *height* and *width* attributes . If height and width are not set, the page might flicker when video loads.
- The <*source*>element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.
- The text between the <*video*> and </*video*> tags will only be displayed in browsers that do not support the <*video*> element.
- HTML5 defines DOM methods, properties, and events for the <*video*>element.

The audio Element

- The HTML5 <audio>element specifies a standard way to embed audio in a web page.
- There are 3 supported audio format in HTML5 : MP3, OGG, WAV
- The browser support for the different formats is:

Browser	MP3	WAV	Ogg
Internet Explorer	Yes	No	No
Chrome	Yes	Yes	Yes
FireFox	Yes	Yes	Yes
Safari	Yes	Yes	No
Opera	Yes	Yes	Yes

- HTML audio-media types

File Format	Media Types

Mp3	audio/mpeg
Ogg	audio/ogg
Wav	audio/wav

Example:

```
<!DOCTYPE html>
<html>
        <title> Audio Element</title>
<body>
<audio controls autoplay>
        <source src="horse.ogg" type="audio/ogg">
        <source src="horse.mp3" type="audio/mpeg">
        Your browser does not support the audio element.
</audio>
</body>
</html>
```

How it works:

- The *controls* attribute adds audio controls, like play, pause, and volume.
- The <*source*>element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

Some attributes/methods/properties used in audio/video element:

There are many attributes/properties that are used in audio/video element . Some of them are given below:

- *autoplay* – it start playing audio/video as soon as it is loaded.
- *audioTracks* – returns an Audio TrackList object representing available audio tracks.
- *buffered* – returns a TimeRanges object representing the buffered parts of the audio/video.
- *currentSrc* – returns the URL of the current audio/video.
- *currentTime* – sets or returns the current playback position in the audio/video(in seconds).
- *duration* – returns the length of the current audio/video(in seconds).
- *paused* – returns whether the audio/video is paused or not.
- *muted* – sets or returns whether the audio/video is muted or not.
- *play(event)* – fires when the audio/video has been started or is no longer paused.
- *loop* – sets or returns whether the audio/video should start over again when finished.

Image mapping In HTML

- To create an image as a map, ***<map>*** tag is used and is linked to the image by using the *name* attribute.
- The ***name*** attribute must have same value to ***usemap*** attribute.

Example:

```
<!DOCTYPE html>
<html>
<head>
        <title>Image Mapping</title>
```

```
</head>
<body>
        <p>Image map is to define map with clickable areas!</p>
        <img src="test1.jpg" height="auto" width="auto" usemap="#img-maping" alt="Image Here">
        <map name="img-maping">
                <area shape="circle" coords="53,40,30" href="htmlForm.html" target="_blank">
                <area shape="rect" coords="110,28,170,69" href="htmlForm.html" target="_blank">
                <area shape="poly" coords="86,86,64,111,88,134,111,110" href="htmlForm.html"
target="_blank">
        </map>
</body>
</html>
```

***elements* and *attribute* used on above example:**

- ***<area>*** - this element is used to define clickable area on image used.
- ***shape*** – attribute define the shape of area you want.
  - circle – circular region
  - rect – rectangular region
  - poly – polygonal region
  - default – entire region
- ***coords*** – attribute define coordinates to be able to place clickable on the image.
  - For circle: “100,50,30” where (100,100) is center point and (30) is radius .
  - For rect: “100,100,200,150” (100,100) is one point of diagonal and (200,150) is next one point.

Chapter 3
Cascading Style Sheets

Some Old Questions:

a) Why CSS is used in web designing ? Explain CSS Box Model and its properties used .

b) What are the main purposes of id and class selectors? Explain with respective examples.

c) What is the function of selectors in CSS? Explain all sorts of selectors used in CSS with an example.

d) What do you mean by CSS? What is the advantage of external style sheets over inline and document-level style sheets?

e) Describe different properties and property values associated with fonts, lists , colors and images.

f) Why CSS is used in Web designing? Explain different levels of styles sheets with example. Also write advantages and disadvantages of each.

g) Short Notes On:

   i. Div and Span
   ii. Display and Visibility
   iii. Class selector and pseudo class

Introduction and Levels of style sheets:

- CSS stands for “Cascading Style Sheets”.
- Cascading Style Sheets is used to format the layout of web pages.
- CSS helps web developers to create uniform look across several pages on websites.
- Using CSS makes task easier to define common layout across the pages.
- CSS is specially used to style pages, including the design, layout and variations in display for different devices and screen sizes. .

There are three levels of style sheets:

(The style specification formats : The format of a style specification depends on the level of style sheet .)

1. **Inline**
   - is specified for a specific occurrence of a tag and apply only to that tag.
   - appears in the ***tag*** itself.
   - is fine-grain style, which defeats the purpose of style sheets - uniform style.

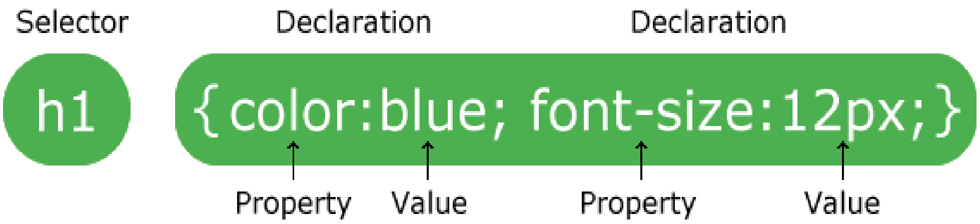
2. **Document-level** style sheets
   - apply to the whole document in which they appear
   - appear in the ***head*** of the document

3. **External** style sheets
   - can be applied to any number of documents
   - are in separate files, potentially on any server on the Internet
   - use a <*link*> tag to let the browser fetch and use an external style sheet file
   - <link rel = "stylesheet" type = "text/css" href = "style.css"></link>
   - The file should not contain any html tags.
   - The style sheet file must be saved with a .css extension.

   Note: So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

CSS Syntax:

- A CSS rule-set consists of a selector and a declaration block:



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.
- In the above example all <h1> elements will be of font color blue, with font size 12px.

Example:

```
<!DOCTYPE html>
<html>
<head>
//Internal level css
<style>
body {
                        background-color: linen;
}
h1 {
                       color: maroon;
                        margin-left: 40px;
}
</style>
//External level css
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
   <h1>This is a heading</h1>
   <p>This is a paragraph.</p>
   <p style="color: red;">This is also paragraph</p>        //Inline Level css
</body>
</html>
/*(style.css )This is an external file named style.css*/
   h1{ font-size: 25px;}
```

Note:

Do not add a space between the property value and the unit (such as margin-left: 25 px;) . The correct way is : margin-left: 25px;

CSS Selectors

- CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

  Types of selectors:

  1. **The element selectors**
     - The element selector selects elements based on the element name.
     - The simplest selector form is a single element name ,such as *h1*.
     - The property values in the rule apply to all occurrences of the named element.
     - The selector could be a list of element names separated by commas (*h1,h2,h3,……).*

  2. **Class selectors**
     - The class selector selects elements with a specific class attribute.
     - To select elements with a specific class, write a period (.) character, followed by the name of the class.
     - Class selectors are used to allow different occurrence of the same tag to use different style specifications.

  3. **Id selectors**

     - The id selector uses the *id* attribute of an HTML element to select a specific element.

- The id of an element should be unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

4. Universal selector

- CSS Universal selector selects any type of elements in HTML page.
- CSS Universal selector is defined by using asterisk (i.e. *).
- CSS Universal selector is also followed by selector.
- Universal selector is useful to style all elements in HTML pages or used to style all elements with in element.

5. Pseudo Classes

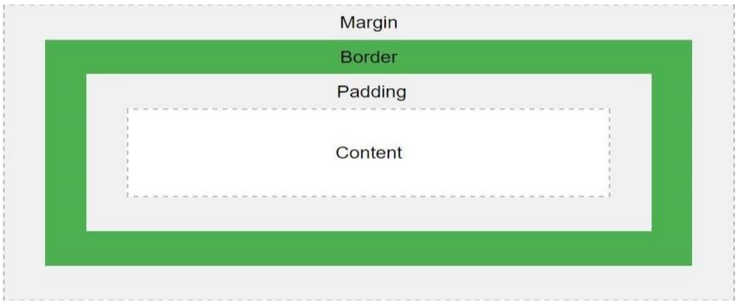
- Sometime we need to apply style when something happen , rather than because the target element simply exists, i.e. specify by pseudo classes.
-

Example:

```
/* Single '*' is used to set up for all body elements */
*{
    color: blue;  /*defines blue color for all font*/
   background-color: green; /*define green background color */
    font-size: 20px;   /*defines font size for all font*/
}
/*Following use of asterisk is used to style all elements of div element*/
div *{
    color: yellow; /*define yellow color to all fonts with in div elements*/
    font-size: 30px; /*define font size to all fonts with in div elements*/
}
```

Box Model

- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content

- **Margin** - Clears an area outside the border. The margin is transparent .

The box model allows us to add a border around elements, and to define space between elements.

CSS Margin:

- The CSS *margin* properties are used to create space around elements, outside of any defined borders.
- CSS has properties for specifying the margin for each side of an element.
  - margin-top
  - margin-right
  - margin-bottom
  - margin-left

All the margin properties can have the following values:

- *auto* - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- *%* - specifies a margin in % of the width of the containing element
- *inherit* - specifies that the margin should be inherited from the parent element

Example:

If the margin property has four values:

margin: 25px 50px 75px 100px;

- top margin is 25px
- right margin is 50px
- bottom margin is 75px
- left margin is 100px

If the margin property has three values:

margin: 25px 50px 75px;

- top margin is 25px
- right and left margins are 50px
- bottom margin is 75px

If the margin property has two values:

margin: 25px 50px;

- top and bottom margins are 25px
- right and left margins are 50px

If the margin property has one value:

margin: 25px;

- all four margins are 25px

Note:

- The *auto* value – when the auto value is set to margin property , specified width took up first and then remaining space will be split equally between left and right margin from center of container .
- The *inherit* value – lets the left margin of the elements be inherited from parent element.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
    div {
            border: 1px solid red;
            margin-left: 100px;
    }
    p.demo {
            margin-left: inherit;
    }
    div.demo1 {
            width:300px;
            margin: auto;
            border: 1px solid red;
    }
</style>
</head>
<body>
    <p>You can set margin horizontal< p>
    <div class="demo1">
            This div will be horizontally centered because it has margin: auto;
    </div>
    <h2>Use of the inherit value</h2>
            <p>Let the left margin be inherited from the parent element:</p>
    <div>
            <p class="demo">This paragraph has an inherited left margin (from the
    div element).</p>
    </div>
</body>
</html>
```

CSS Padding:

- The CSS padding generates space around the elements content, inside of any defined border.
- CSS has properties for specifying the padding for each side of an element.
    - padding-top
    - padding-right
    - padding-bottom
    - padding-left

All the padding properties can have the following values:

- *auto* - the browser calculates the margin

- *length* - specifies a padding in px, pt, cm, etc.
- *%* - specifies a padding in % of the width of the containing element
- *inherit* - specifies that the padding should be inherited from the parent element

Example:

If the padding property has four values:

padding: 25px 50px 75px 100px;

- top padding is 25px
- right padding is 50px
- bottom padding is 75px
- left padding is 100px

If the padding property has three values:

padding: 25px 50px 75px;

- top padding is 25px
- right and left paddings are 50px
- bottom padding is 75px

If the padding property has two values:

padding: 25px 50px;

- top and bottom paddings are 25px
- right and left paddings are 50px

If the padding property has one value:

padding: 25px;

- all four paddings are 25px

Note:

- The width property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element.
- So, if an element has a specified width, the padding added to that element will be added to the total width of the element.

CSS Border

- The CSS border properties allow you to specify the style, width, and color of an element's border.
  Some border related properties are given below:
  - **border-style** – specify what kind of border to display (values are*(dotted, dashed, solid, double, groove, ridge, inset, outset, none, hidden)*. Also can have one to four value(*border-top-style, border-right-style, border-bottom-style, border-left-style*).

- **border-width** – specify width of the four borders.
- **border-color** – specify color of the four borders. Also can have one to four value (*top*, *right*, *bottom*, *left*).
- **border** – this property is used as shorthand property of *border-width, border-style and border-color*.
- **border-radius** – is used to add rounded borders to an element.

Example:

```
<!DOCTYPE html>
<html><head>
<style>
          p.one {
                        border-style: solid;
                         border-color: red;  //This not support if not border-style
          }
          p.two {
                        border-style: solid;
                        border-color: green;
                         border-radius: 5px;
          }
          p.three {
                        border-style: solid;
                         border-color: red green blue yellow;
          }
p.four{
          border: 1px dashed blue;
          }
</style></head>
<body>
          <h2>The border-color Property</h2>
          <p class="one">A solid red border</p>
          <p class="two">A solid green rounded border</p>
          <p class="three">A solid multicolor border</p>
          <p class="four">This is shorthand property:</p>
</body>
</html>
```

CSS Visibility

- Visibility property effect on the appearance of an element.
- Visibility property specifies whether or not an element visible.
- When you use value hidden , the hidden element take space on the page.
- Visibility property values are
  - **visible** – default value , element is visible
  - **hidden** – the element is hidden but still take space
  - **collapse** – only used in table , collapse the borders
  - **inherit** – specify that a property should inherit its value from its parent element
  - **initial** – sets this property to its default value

Example:

```
<!DOCTYPE html>
<html>
<head>
   <style>
          h2.a {
                  visibility: visible;
                  }
          h2.b {
                  visibility: hidden;
          }
  </style>
</head>
<body>
  <h1>The visibility Property</h1>
  <h2 class="a">This heading is visible</h2>
  <h2 class="b">This heading is hidden</h2>
<p>Notice that the hidden heading still takes up space on the page.</p>
</body>    </html>
```

CSS Display

- The display property is an important property that is used for controlling layout.
- The display property specifies if/how an element is displayed.
- As you saw that every HTML element has a default display value depending on what types of element it is.
- The default display value for most elements is either block or inline.
- A block-level element always starts on a new line and takes up the full width available(*eg: <div>, <h1>….<h6>, <header>, <footer>, and <section>*).
- An inline element does not start on a new line and only takes up as much width as necessary(*eg: <span>, <img>, and <a>*).
- Some values of display property are as follows:
  - **display**: *none* – is used to hide and remove an element from document layout.
  - **display**: inline – is used to display in inline format.
  - **display**: block – is used to display in block format.

CSS Backgrounds

- The CSS backgrounds are specified for background effects on elements.
- The CSS background properties are:
  - **background-color** – specifies background color of an element.
  - **background-image** – specifies background image of an element(*background-image*: url("karnali.jpg"). ***Note*** *– if necessary give path*.
  - **background-attachment** - property sets whether a background image scrolls with the rest of the page, or is fixed. (Values are: *scroll, fixed, inherit, initial and local*)
  - **background-position** - sets the starting position of a background image.(Values are: *left top, left center, left bottom, right top, right center, right bottom, center top, center bottom, and center center*)
  - **background-repeat** - sets if/how a background image will be repeated. (Values are: *no-repeat, repeat-y, repeat-x, space, and round*)
  - **background-size** - specifies the size of the background images.(Values are: *auto, cover, contain, length, percentage, initial, and inherit*)

- **background-origin** - specifies the origin position (the background positioning area) of a background image.(Values are: *padding-box, content-box, border-box, inherit and initial*)
- **background-clip -** defines how far the background (color or image) should extend within an element. (Values are: *padding-box, content-box, border-box, inherit and initial*)
- **background-blend-mode** - defines the blending mode of each background layer (color and/or image).(Values are: *normal, multiply, screen, overlay, darken, lighten, color-dodge, saturation, color, luminosity;*)

  Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
#myDIV {
            width: 200px;
             height: 200px;
            background-repeat: no-repeat, repeat;
            background-image: url("bas2.jpg"), url("karnali.jpg");
            background-blend-mode: lighten;
  }
</style>
</head>
<body>
            <div id="myDIV"></div>
            <p><b>Note:</b> Edge/Internet Explorer do not support the
  background-blend-mode property.</p>
</body>
</html>
```

CSS Colors

- Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.
- You can add color to text, border and background.
- There are different method to assign colors. Which are given below:

  - **RGB Value** - In HTML, a color can be specified as an RGB value, using this formula: **rgb(*red, green, blue*).** Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.
  - **HEX Value** - In HTML, a color can be specified using a hexadecimal value in the form: ***#rrggbb*** Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).
  - **HSL Value** - In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form: **hsl(*hue, saturation, lightness*)** Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue. Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color. Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white.

- **RGBA –** RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color. An RGBA color value is specified with: **rgba (*red, green, blue, alpha*)** The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):
- **HSLA -** are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

CSS Text Properties

- **color** – is defined to set color for text.
- **text-decoration** – is used to set or remove decorations from text.(Values are: none, underline, overline and line-through)
- **text-align** – is used to set horizontal text alignment.(Values are: left, center, right and justify)
- **text-transform** – is used to specify uppercase and lowercase letters in text.(Values are: uppercase, lowercase and capitalize)
- **text-indent** – is used to specify indentation of the first line of a text.
- **line-height** – is used to define space between lines.
- **direction** – is used to change the text direction of an element. (Value is *rtl* i.e. right to left)
- **word-spacing –** is used to specify the space between the words in a text.
- **text-shadow** – add shadow to text.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
        .changeColor{
                color: green;
        }
        .txtAlignment{
                text-align: center;
        }
        a{
                text-decoration: none;   /*remove decoration to text*/
        }
        .uppercase{
                text-transform: uppercase;
        }
        .txtIndent{
                 text-indent: 50px;
        }
        h2 {
                letter-spacing: 7px;    /*Also can use -ve number.*/
        }
        .txtShadow{
                text-shadow: 3px 2px red;
        }
</style>
</head>
<body>
```

```
                <h1 class="changeColor">This is heading 1</h1>
                <h1 class="txtAlignment">You can align as you like</h1>
                <a href="https://www.google.com">Hello</a>
                <p class="uppercase">You Can use other Transform.</p>
                <p class="txtIndent"> It start text line after 50px indentation.</p>
                <h2>This is heading 2</h2>
                <h1 class="txtShadow">Text-shadow effect</h1>
</body>
</html>
```

CSS max-width property

- The max-width is used to set the maximum width of an element.
- The max-width can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).
- The max-width property handle the browser if the content of element is larger then browser.

Example:

```
<html>
<head>
<style>
div {
  max-width: 500px;
                              height: 100px;
                             background-color: powderblue;
}
</style>
</head>
<body>
<h2>Set the max-width of an element</h2>
<div></div>
<p>Resize the browser window to see the effect.</p>
</body>
</html>
```

CSS Layout – The position Property

- specifies the type of positioning method used for an element (*static*, *relative, fixed*, *absolute* or *sticky*).
- position ***static*** – this is default value .
- position ***relative*** – relative to its normal position.
- position ***fixed*** – is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
- position ***absolute*** – is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- position ***sticky*** – is positioned based on the user's scroll position .A sticky element toggles between relative and fixed , depending on the scroll position.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.shift{
                             top: 100px;
```

```
                        padding-top: 25px;
            }
            div.relative {
                        position: relative;
                        width: 400px;
                        height: 200px;
                        border: 3px solid #73AD21;
            }
            div.absolute {
                         position: absolute;
                         top: 80px;
                         right: 0px;
                        width: 200px;
                        height: 100px;
                        border: 3px solid #73AD21;
            }
            img {
                        position: absolute;
                        left: 0px;
                        top: 0px;
                        z-index: -1;
            }
            </style>
            </head>
            <body>
                  <h1>This is a heading</h1>
                  <img src="bas2.jpg" width="300" height="140">
                  <p>Because the image has a z-index of -1, it will be placed behind the text.</p>
                  <div class="shift"><h2>position: absolute;</h2>
                  <p>An element with position: absolute; is positioned relative to the nearest      positioned
            ancestor (instead of positioned relative to the viewport, like    fixed):</p>
                  <div class="relative">This div element has position: relative;
                  <div class="absolute">This div element has position: absolute;</div>
            </div>
            </div>
            </body>
            </html>
```

<div> and <span>:

div Element:

- *div* element is a container of content flow(HTML) also called block element.
- *<div>* tag defines a section or division in an HTML document.
- *<div>* tag has no effect on content and layout until styled(css).
- <div> tag has full width when defined with in HTML document.

Syntax:

```
<div>
      <div>
            <div>I am Div Tag.</div>
            <p>I am Paragraph.</p>
      </div>
</div>
```

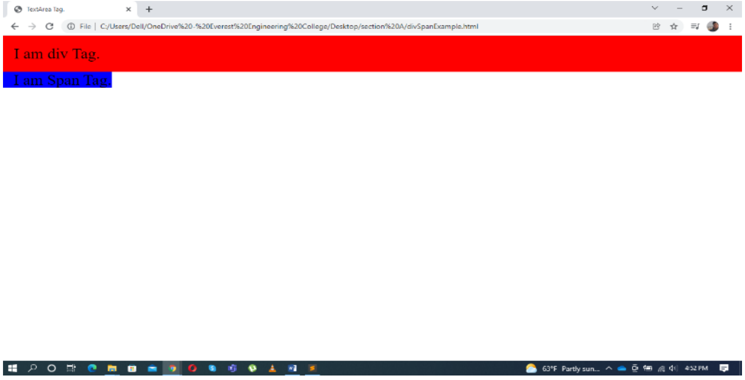
span Element:

- span element is an inline container for inline elements and content.
- <span> tag is used ,occur in pair i.e. <span>….</span>.
- <span> tag very much similar to <div> tag but<div> is block-level and <span> tag is inline tag.

  Syntax:

  <span> I am Span element.</span>

Example: Include both ***div*** and ***span***.



TextArea Tag.
File | C:/Users/Dell/OneDrive%20-%20Everest%20Engineering%20College/Desktop/section%20A/divSpanExample.html

I am div Tag.

I am Span Tag.

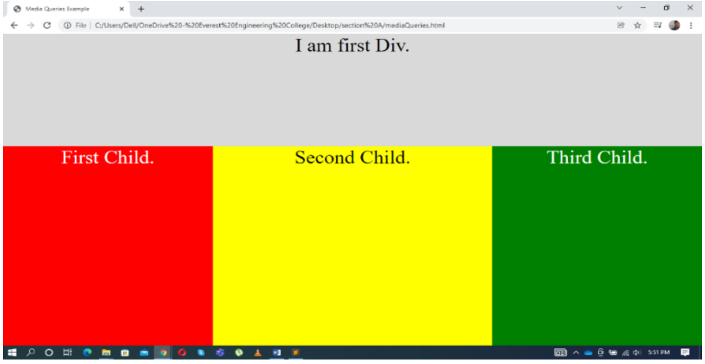
Media Queries:

- CSS2 first introduced media types(*@media* rule), made possible to define different style rules for different media types.
- Here, every individual devices(computer screens, printers , handheld devices , television-type devices and so on) have one set of style rules.
- Unfortunately ,except print media type , never got a lot of support by other devices.
- Now, CSS3 extended CSS2 media types idea : Instead of looking for a device type, they look at the capability of the device.
- Media queries can be used to check many things, such as:
  - ➢ Width and height of the viewport
  - ➢ Width and height of the device
  - ➢ Orientation(is the tablet/phone in landscape or portrait mode?)
  - ➢ Resolution

CSS3 Media Types:

Value	**Description**
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screen readers that “reads” the page out loud

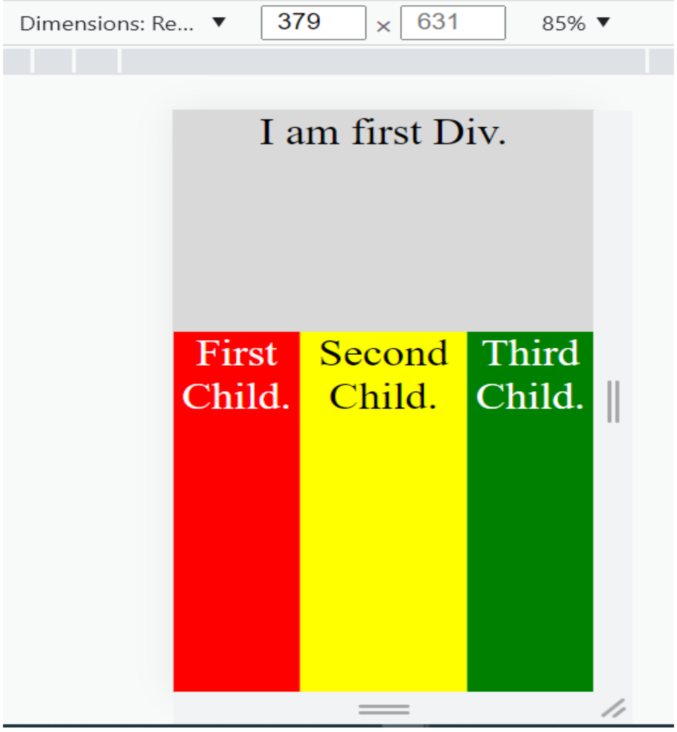
Example:



I am first Div.

First Child. Second Child. Third Child.

Above image is a design according to PC screen but if load above design to mobile device or other then design be like:



I am first Div.

First Child. Second Child. Third Child.

Which is not responsive to other devices. So, design must be relative to screens/devices . To made design responsive , use media queries.

Program

----- mediaQueries.html -----

```
<!DOCTYPE html>
<html>
<head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <title>Media Queries Example</title>
      <link rel="stylesheet" type="text/css" href="./mediaQueries.css">
</head>
<body>
      <div class="firstDiv">
```

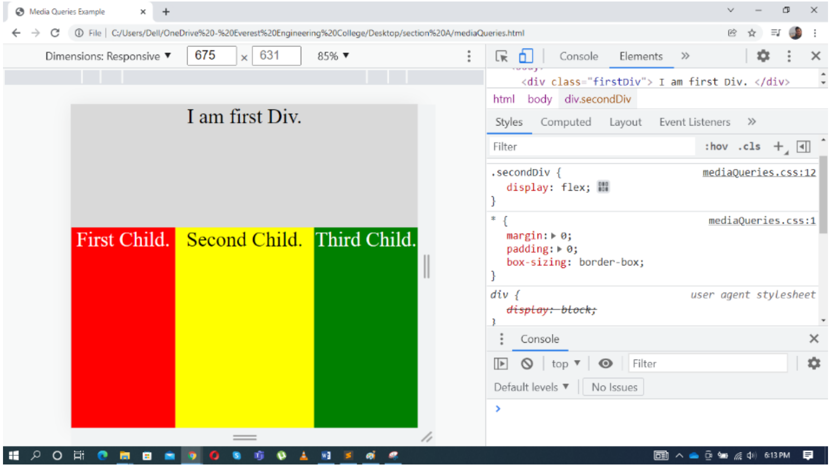
```
            I am first Div.
    </div>
    <div class="secondDiv">
            <div class="firstChild">First Child.</div>
            <div class="secondChild">Second Child.</div>
            <div class="thirdChild">Third Child.</div>
    </div>
</body>
</html>
```

----- mediaQueries.css -----

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
.firstDiv{
    height: 240px;
    background-color: #d9d9d9;
    font-size: 40px;
    text-align: center;
}
.secondDiv{
    display: flex;
}
.firstChild{
    height: 427.2px;
    width: 30%;
    background-color: red;
    color: #FFFFFF;
    font-size: 40px;
    text-align: center;
}
.secondChild{
    height: 427.2px;
    width: 40%;
    font-size: 40px;
    background-color: yellow;
    text-align: center;
}
.thirdChild{
    height: 427.2px;
    width: 30%;
    background-color: green;
    color: #FFFFFF;
    font-size: 40px;
    text-align: center;
}
@media screen and (max-width: 660px){
    .firstDiv{
            height: 62px;
```

```
        }
        .secondDiv{
                display: flex;
                flex-direction: column;
        }
        .firstChild{
                width: 100%;
                height: 200px;
        }
        .secondChild{
                width: 100%;
                height: 200px;
        }
        .thirdChild{
                width: 100%;
                height: 200px;
        }
}
```

Output: **viewport greater than 660px:**



C:/Users/Dell/OneDrive%20-%20Everest%20Engineering%20College/Desktop/section%20A/mediaQueries.html

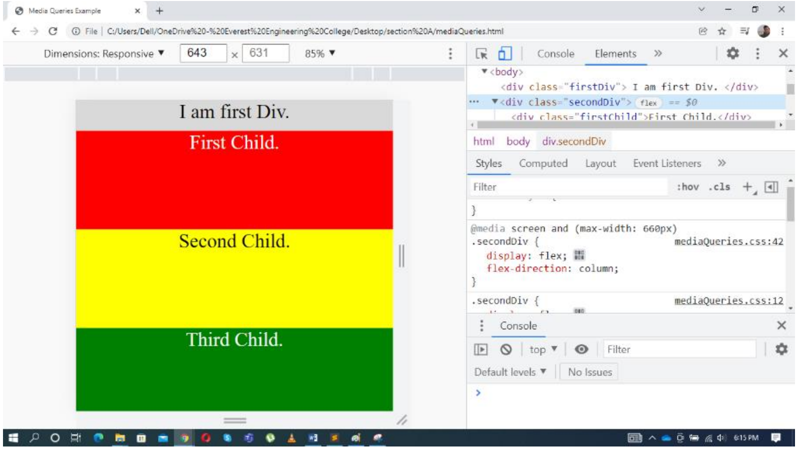
Dimensions: Responsive 675 × 631 85%

I am first Div.

First Child. Second Child. Third Child.

```
<div class="firstDiv"> I am first Div. </div>

.secondDiv {                          mediaQueries.css:12
  display: flex;
}
* {                                    mediaQueries.css:1
  margin: ▸ 0;
  padding: ▸ 0;
  box-sizing: border-box;
}
div {                             user agent stylesheet
  ~~display: block;~~
}
```

viewport less than 660px:



C:/Users/Dell/OneDrive%20-%20Everest%20Engineering%20College/Desktop/section%20A/mediaQueries.html

Dimensions: Responsive 643 × 631 85%

I am first Div.

First Child.

Second Child.

Third Child.

```
▾<body>
   <div class="firstDiv"> I am first Div. </div>
 ▾<div class="secondDiv"> flex == $0
     <div class="firstChild">First Child.</div>

}
@media screen and (max-width: 660px)
.secondDiv {                          mediaQueries.css:42
  display: flex;
  flex-direction: column;
}
.secondDiv {                          mediaQueries.css:12
```

<meta> Tag:

- Defines metadata about an HTML document. Where metadata means data / information about data.
- Used to specify character set, page description , keywords, author of document, and viewport settings.
- Used by browsers define, how to display content and relative pages, search engines(keywords) , and other web services.

  Examples:

  - ➢ Define keywords/description/author

    <meta name= "keywords " content= "HTML,CSS,JavaScript " >
    <meta name= "description " content= " Web Technology Class " >
    <meta name= "author" content= " Santosh Bist" >
  - ➢ Refresh document every 30 second.

    <meta http-equiv = "refresh" content= "30" >
  - ➢ Specifies the character encoding for the HTML document

    <meta charset="utf-8">
  - ➢ Seeting the viewport to make your website look good on all devices.

    <meta name="viewport" content="width=device-width, initial-scale=1">

    width=device-width – sets the width of the page to follow the screen-width of the device.

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

Note: **Do yourself remaining properties like min(),max(),grid , …….**

CHAPTER 4
JavaScript

Some Old Questions:

a) Explain Lexical Structure of JavaScript.

b) Is Java and JavaScript same ? Explain . Also , write a program that illustrate display and visibility properties with JavaScript code.

c) What do you mean by event and event handling ? Write a program that take number from the user and results its factorial after clicked on button "**CALCULATE**" .

d) What do you mean by DOM in JavaScript? How dragging and dropping of XHTML elements is possible in JavaScript?

e) Write JavaScript code to find the factorial of a number requesting a number from user prompt. How Element visibility can be changed using JavaScript?

f) What DOM stands for? Describe DOM model in JavaScript.

g) Can JavaScript be used as server side script? How element visibility can be changed using JavaScript? Explain with example.

h) Write a program to alter the visibility of an image of HTML document using JavaScript.

i) Write a program in JavaScript to validate a form. The user form includes username, phone no. and submit button . On clicking the submit button validate the following:
   i. If the name or phone no. fields are blank.
   ii. If the phone number contains numeric digits only.

j) Write a JavaScript program which validates the user data (name, email, contact no) and displays the success message if validated successfully. Also create the HTML form for entering the data.

k) Create an HTML form that takes user's email address and password. Validate the form using JavaScript when the submit button is clicked. Make sure that the fields are not empth and the email address is a valid one.

l) Explain DOM. Explain Handling event from page load and button elements with an example.

m) Explain constructor in JavaScript with example.

n) What do you mean by DOM in JavaScript? Why is JavaScript called client Side scripting language?

o) How do you move elements, change font styles of the texts and drag and drop elements with JavaScript? Explain with appropriate examples.

p) Write JavaScript code to illustrate element positioning , stacking elements and mouse cursor.

q) Is JavaScript Object Oriented Programming Language? Illustrate your answer with suitable reasons an examples.

r) Different between "==" and "===".

s) Short Notes On:
   i. Explain DOM Hierarchy in JavaScript.
   ii. Type conversion in JavaScript

Overview of JavaScript

- JavaScript is a programming language commonly used in web development.
- JavaScript was originally developed at Netscape by Brendan Eich as a means to add dynamic and interactive elements to websites , was initially named Mocha but soon after was renamed LiveScript .
- LiveScript became a joint venture of Netscape and Sun Microsystems , and its name again was changed and know as JavaScript, in late 1995.
- JavaScript can be divided into three parts: the core , client side , and server side.
- The core part including its operators, expressions , statements , and subprograms, also called heart of the language.
- Client-side JavaScript is a collection of objects that support the control of a browser and interactions with users.
- Server-side JavaScript is a collection of objects that make the language useful on a Web server – for example , to support communication with a database management system .
- But here , we introduce only about client - side JavaScript .
- Client – side JavaScript is an Html embedded scripting language.

Is JavaScript and Java are same ?

Although JavaScript's name seems close to Java , JavaScript and Java are actually very different. One important difference is support for object – orientation programming . JavaScript object model is quite different from that of Java and C++. Also JavaScript does not support the object – oriented software development paradigm.

Some points that differentiate Java and JavaScript are given below:

- **Java** is an object – oriented programming language where **JavaScript** is an object – based programming language.
- **Java** is a strongly typed language , all types are known at compile time, and operand types are checked for compatibility but JavaScript is a weakly typed language where variable need not be declared and are dynamically typed.
- **Java** have classes and define objects but **JavaScript** do not have classes , its objects serve both as objects and as models of objects.
- Object – oriented language support class – based inheritance Where **JavaScript** support a technique that can be used to simulate some of the aspects of inheritance called prototype – based inheritance , is done using prototype objects.

JavaScript Objects

- Objects are collections of properties , which correspond to the members of classes in Java and C++ . Each property is either a data property or a function or method property.
- Data properties appear in two categories: primitive values and references to other objects. Also , variables that refer to objects are often called objects rather then references.
- JavaScript uses nonobject types for some of its simplest types; these nonobject types are called primitives.
- In **JavaScript** almost "everything" is an object .
  - Booleans can be objects (if defined with the new keyword)
  - Numbers can be objects (if defined with the new keyword)
  - Strings can be objects (if defined with the new keyword)
  - Dates are always objects

- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

- Objects are variables too. But object can contain many values.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>
<script>
// Create an object:
var person = {
        firstName: "Ramesh",
        lastName: "Karki",
        age: 20,
        eyeColor: "blue"
};
// Display some data from the object:
document.getElementById("demo").innerHTML =person.firstName+ " is" + person.age + " years old.";
</script>
</body>
</html>
```

General Syntactic Characteristics /Lexical structure

A programming language’s lexical structure specifies set of some basic rules about how a code should be written in it. Rules like what variable names looks like, the delimiter characters for comments, and how one program statement is separated from the next. It is the lowest-level syntax of a language.

i. **Character Set - Unicode** character set is used in JavaScript codes.

ii. **Case Sensitive** – **JavaScript** is case-sensitive language. While , while , WHILE all are different in JavaScript. It means that language keywords, variables, function names, and other identifiers must always be typed with a consistent capitalization of letters.

iii. **Whitespace and Line Breaks - JavaScript** is ignorant to spaces that appear between tokens in codes. Line breaks are also not considered by JS. This lets you freely use any number of spaces and newlines.

iv. **Comment** – There are two types of comment . First one is single line comment (i.e. double forward slash(//)) and second one is multiple line comment(i.e. followed by forward slash and * pair(/* and */).

v. **Identifier and Reserved words** – A JavaScript **identifier** must begin with a letter, an underscore (_), or a dollar sign ($). Digits are not allowed as the first character. The identifiers reserved by the language for its use are known as “**reserved words**”. Example: var, typeof , case , continue etc.

vi. **Optional Semicolon –** Semicolons are used in JavaScript to separate statements from each other. But you can also omit the semicolon between two statements if those statements are written on separate lines.

vii. All JavaScript scripts will be embedded in HTML documents

- Either directly, as in
  <script type = "text/javaScript">
  -- *JavaScript script* - -
  </script>
- Or indirectly, as a file specified in the src attribute of <script>, as in
  <script type = "text/javaScript" src = "myScript.js">
  </script>

viii. Scripts are usually hidden from browsers that do not include JavaScript interpreters by putting then in special comments.

```
<!--
--JavaScript script --
//-->
```

- Also hides it from HTML validators

Browsers and HTML-JavaScript Documents

HTML-JavaScript Documents form different execution method . If HTML document does not include scripts, the browser reads the lines of the document and renders its window according to the tags , attributes , and content it finds. When a JavaScript encountered in the document , the browser uses its JavaScript Interpreter to “execute” the script. The output form of the script becomes the next markup to be rendered and when the end of the script is reached , the browser goes back to reading the HTML document and displaying its content.

There are two different ways to embed JavaScript in an HTML document : **implicitly** and **Explicitly**. In **explicit** embedding , the JavaScript code physically resides in the HTML document. In **implicit** embedding , JavaScript can be placed in its own file, separate from the HTML document.

Uses of JavaScript

- The main objective of JavaScript was to provide programming capability at both server and the client ends of a web connection.
- Client-side JavaScript can serve as an alternative in which computational capability resides on the server and is requested by the client.
- In particular, although server-side software supports file operations , database access, and networking , client-side JavaScript supports none of these.
- JavaScript is used to make interactions with users through form elements, such as buttons menus , they can be used to trigger computations and provide feedback to the user.
- To load data in the background from the server and loaded on to the page side without reloading.
- Client side validation of form elements instead of sending data to server every time.

JavaScript Variables

- In javascript variables are defined using *var* keyword.
- We don’t need to define data type as in C, C++, Java etc.
  Example:

```
var str=5; //Here str is an integer type
var str=5.0; //Here str is float type
var str="hello"; //Here str is string type
```

- As like above we can declare any type variables using *var* keyword.

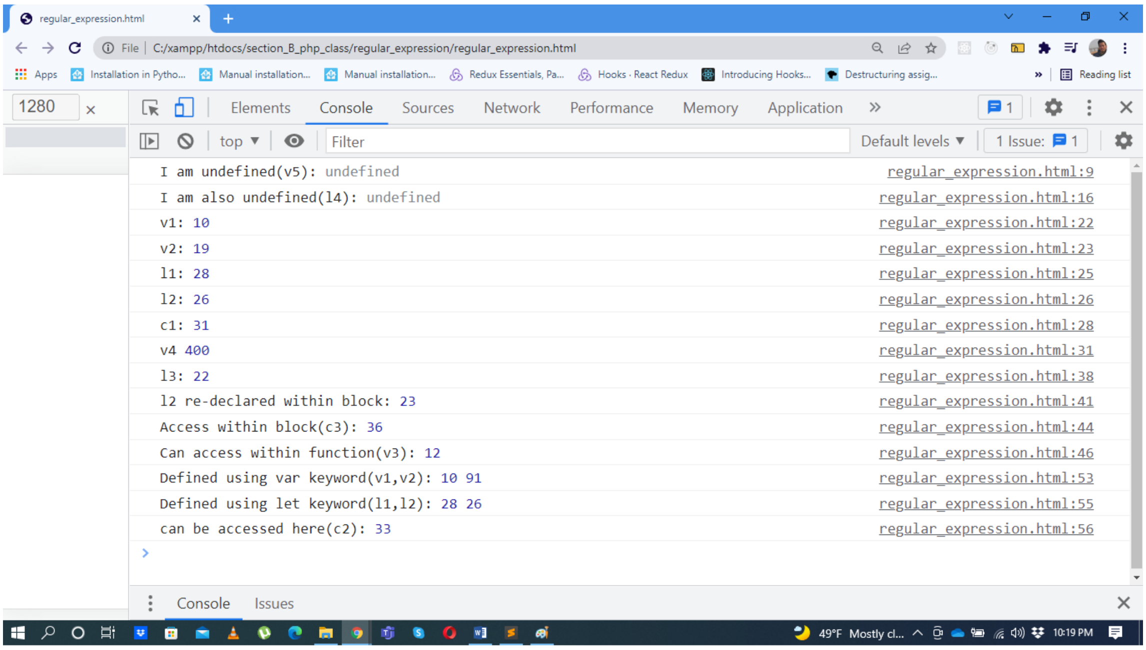
Different between *var* , *let* and *const* keyword.

var	***let***	***const***
Functional scope	Block scope	Block scope
Can be updated and re-declared.	Can be updated but cannot be re-declared.	Cannot be updated or re-declared
Can be declared without initialization.	Can be declared without initialization.	Cannot be declared without initialization.
Can be accessed without initialization as its default value is undefined.	Can be accessed without initialization as its default value is undefined.	Cannot be accessed.

Example:

```
<script type="text/javascript">
        //first var keyword to define/declare variables
        //declared globally, accessible everywhere
                var v1 = 10;
                var v2;
                v2 = 19;
                var v4 =100;
                var v5;
                console.log("I am undefined(v5):",v5);
        //define/declare variables using let keyword
                let l1 = 28;
                let l2 ;
                l2= 26;
                let l4;
                console.log("I am also undefined(l4):",l4);
        //define/declare variables using const keyword
                // const c1; //not allowed
                const c1 = 31;
                const c2 = 33;
        function letsCheckOut() {
                console.log("v1:",v1);
                console.log("v2:",v2);

                console.log("l1:",l1);
                console.log("l2:",l2)
                console.log("c1:",c1);

                v4 = 400; //allowed
                console.log("v4",v4);
                // c2 = 34 //not allowed
                // console.log("c2:",c2);

                if(true){
                        var v3 = 12; //local variable(can't access outside function)
                        let l3 = 22; //access with in block
                        console.log("l3:",l3);
                        let l2 = 23;//re-declared with in different block
                        console.log("l2 re-declared within block:",l2);

                        const c3 = 36;
                        console.log("Access within block(c3):",c3);
                }
                console.log("Can access within function(v3):",v3);
                // console.log("can't access out of block(l3):",l3);
                // console.log("can't access out of block(c3):",c3);
        }
        letsCheckOut();
        var v2 = 91; // can be re-declared
        console.log("Defined using var keyword(v1,v2):",v1,v2);
        // let l2 = 62; // can't be re-declared
        console.log("Defined using let keyword(l1,l2):",l1,l2);
        console.log("can be accessed here(c2):",c2);
</script>
```

Output:



```
I am undefined(v5): undefined                         regular_expression.html:9
I am also undefined(l4): undefined                    regular_expression.html:16
v1: 10                                                regular_expression.html:22
v2: 19                                                regular_expression.html:23
l1: 28                                                regular_expression.html:25
l2: 26                                                regular_expression.html:26
c1: 31                                                regular_expression.html:28
v4 400                                                regular_expression.html:31
l3: 22                                                regular_expression.html:38
l2 re-declared within block: 23                       regular_expression.html:41
Access within block(c3): 36                           regular_expression.html:44
Can access within function(v3): 12                    regular_expression.html:46
Defined using var keyword(v1,v2): 10 91               regular_expression.html:53
Defined using let keyword(l1,l2): 28 26               regular_expression.html:55
can be accessed here(c2): 33                          regular_expression.html:56
>
```

Primitives and Operations

JavaScript have five primitive types: Number, String, Boolean, Undefined, and Null.

iii. **Numeric Literal:** We don't need to define numeric values types (i.e. int, float, etc). The Number type values are represented internally in double-precision floating-point form. Literals numbers in a script can have the forms of either integer or floating-point values. Integer literals are strings of digits. Floating-point literals can have decimal points. exponents, or both.

iv. **String literal:** A string literal is a sequence of zero or more characters defined within single quotes(') Or double quotes ("). String Literals can include characters specified with escape sequences, such as \n and \t . The backslash (\ ) escape character turns special characters into string characters.

Code	Results	description
\'	'	Single-quote
\"	"	Double-quote
\\	\	backslash
\b	Backspace	
\f	Form feed	
\n	New line	
\r	Carriage return	
\t	Horizontal Tabulator	
\v	Vertical Tabulator	

The sequence \" inserts a double-quote in string.

Example:

```
<html>
<body>
    <h2>JavaScript Strings</h2>
    <p>The escape sequence \\ inserts a backslash in a string.</p>
    <p id="demo"></p>
    <p>The escape sequence \" inserts a double-quote in a string.</p>
    <p id="demo1"></p>
    <p>The escape sequence \' inserts a single-quote in a string.</p>
    <p id="demo2"></p>
<script>
    var x = "The character \\ is called backslash.";
    document.getElementById("demo").innerHTML = x;

    var z = "We are the so-called \"Vikings\" from the north.";
    document.getElementById("demo").innerHTML = z;

    var y = 'It\'s alright.';
    document.getElementById("demo2").innerHTML = y;

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

Here, string can be defined as an objects with the keywords *new*.

Example:

```
<html>
<body>
    <p id="demo"></p>
<script>
    var x = "Rajesh Hamal";        // x is a string
    var y = new String("Rajesh Hamal");   // y is an object
    document.getElementById("demo").innerHTML=typeof  x + "<br> "  + typeof  y;
</script>
</body>
</html>
```

Strings Method:

In JavaScript length of a string is found with build-in ***length*** property. Some methods and properties are:

- **indexOf()** – this method returns the index of (the position of) the first occurrence of a specified text in a string.
- **lastIndexOf() –** returns the index of the last occurrence of a specified text in a string.(**Both indexOf()**, and **lastIndexOf ()**return -1 if the text is not found).
- **search() –** method searches a string for a specified value and returns the position of the match.
- **slice()** – extracts a part of a string and returns the extracted part in a new string. The method takes 2 parameters: the start position, and the end position (end not included).
- **charAt** – returns the character from the specified index. Characters in a string are indexed from left to right. The index of the first character is 0, and the index of the last character in a string, called ***stringName,*** is *stringName.length* – 1.
- **charCodeAt()** – returns Unicode of the character at the specified index in a string.
- **concat()** – combine the text of two string and returns new string.
- **replace()** – replaces a specified value with another value in a string. By default, the replace() function replaces **only the first** match.
- **trim()** – is used to remove white space from both the end of strings.
- **trimLeft()** – is used to remove white space from the start of given string.
- **trimRight()** – is used to remove white spaces from the end of the given string.
- **split()** – is used to splits a String object into an array of strings by separating the string into substrings.

  syntax: *string.split([separator],[limit]);*

  separator – specifies the character to use for separating the string. If *separator* is omitted, the array returned contains one element consisting of the entire string.

  limit – integer specifying a limit on the number of splits to be found.
- **toLowerCase**() – returns the calling string value converted to lower case.
- **toUpperCase**() – returns the calling string value converted to upper case.
- **toString()** – is used to returns given string itself. *(**Syntax**: string.toString())*

Example:

```
<html>
<body>
<h2>JavaScript String Methods</h2>
        <p id="demo"></p>
        <p id="demo1"></p>
        <p id="demo2"></p>
        <p id="demo3"></p>
        <p id="demo4"></p>
        <p id="demo5"></p>
        <button onclick="myFunction()">Replace</button>
        <p id="forButton">Please visit Microsoft!</p>
        <button onclick="myFunction1()">Trim</button>
        <button onclick="myFunction2()">To lower</button>
        <button onclick="myFunction3()">To upper</button>
        <button onclick="myFunction4()">Try it</button>
        <p id="demo6"></p>
        <p id="demo7"></p>
<script>
        //illustrating indexOf() method
        var str = "Please locate where 'locate' occurs!";
        var pos = str.indexOf("locate");
        document.getElementById("demo").innerHTML = pos;

        //illustrating lastIndexOf() method
        var str1 = "Please locate where 'locate' occurs!";
        var pos1 = str1.lastIndexOf("locate");
        document.getElementById("demo1").innerHTML = pos1;

        //illustrating search() method
        var str2 = "Please locate where 'locate' occurs!";
        var pos2 = str2.search("locate");
        document.getElementById("demo2").innerHTML = pos2;

        //illustrating slice() method
        var str3 = "Apple, Banana, Kiwi";
        var res = str3.slice(7,13);   //extracts position 7 to 13
        document.getElementById("demo3").innerHTML = res;

        //the position is counted from end of the string, if -ve
        var res1= str3.slice(-12,-6);
        document.getElementById("demo4").innerHTML = res1;

        //the method will slice out the rest of string
        var res2= str3.slice(7);
        document.getElementById("demo5").innerHTML = res2;

        //illustrating replace() method
        function myFunction() {
         var strLine = document.getElementById("forButton") .innerHTML;//replaces only
        first match(case-sensitive too)
         var txt = strLine.replace("Microsoft","our website");
        document.getElementById("forButton ").innerHTML = txt;
        }

        //illustrating trim() method
        function myFunction1(){
        var str4="                    Welcome";
        alert(str4.trim());
        }
        //illustrating toLowerCase() method
        function myFunction2(){
        var str5="Hello World";
        alert(str5.toLowerCase());
        }
        //illustrating toUpperCase() method
        function myFunction3(){
        var str6="Hello World";
```

```
                alert(str5.toUpperCase());  }

                //illustrating concat() method
                var text1 = "Hello";
                var text2 = "World!";
                var text3 = text1.concat(" ",text2);
                document.getElementById("demo6").innerHTML = text3;
        </script>
        </body>
        </html>
```

Example: (charAt(), charCodeAt(), and split())

```
<!DOCTYPE html>
<html>
<head>
         <title></title>
</head>
<body>
<script type="text/javascript">
        function func()    {
                var str = 'Have a good Day.'+"<br>";
                var array = str.split(" ");
                 document.write(array);
        }
        func(); //function called
        function Function()    {
                var str = 'Have a good Day.';
                var array = str.split(" ",3);  //having limit too
                 document.write(array); //(Have,a,good
        }
        Function(); //function called

        //Illustrating charAt() method
        var str = new String( "This is string" );
        document.writeln("<br />str.charAt(0) is:" + str.charAt(0));
        document.writeln("<br />str.charAt(1) is:" + str.charAt(1));
        document.writeln("<br />str.charAt(2) is:" + str.charAt(2));
        document.writeln("<br />str.charAt(3) is:" + str.charAt(3));
        document.writeln("<br />str.charAt(4) is:" + str.charAt(4));
        document.writeln("<br />str.charAt(5) is:" + str.charAt(5));
        //Illustrating charCodeAt() method
        var str1 = new String( "This is string" );
         document.write("<br />str1.charCodeAt(0) is:" + str1.charCodeAt(0));
        document.write("<br />str1.charCodeAt(1) is:" + str1.charCodeAt(1));
        document.write("<br />str1.charCodeAt(2) is:" + str1.charCodeAt(2));
        document.write("<br />str1.charCodeAt(3) is:" + str1.charCodeAt(3));
        document.write("<br />str1.charCodeAt(4) is:" + str1.charCodeAt(4));
        document.write("<br />str1.charCodeAt(5) is:" + str1.charCodeAt(5));
  </script>
</body>
</html>
```

v. **Others Primitive Types**

- The three other primitive types are Null, Undefined and Boolean.

- The reserved word *null* is the only the value of type Null, which indicates no value.
- A variable is *null* if it has not been explicitly declared or assigned a value, also that will cause runtime error.
- The only value of type Undefined is *undefined* i.e. reserved.
- If a variable has been explicitly declared, but not assigned a value, it has the value *undefined*.
- The only values of type Boolean are *true* and *false*.
- These values are usually computed as the result of evaluating a relational or Boolean expression.

vi. **Numeric Operators**

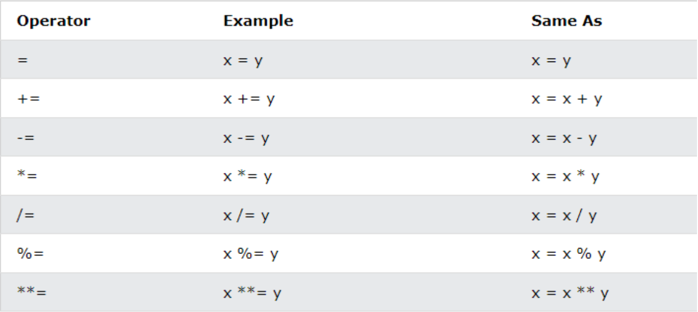
- In JavaScript , numeric operators are similar to operators used in other programming language .
- But sign(+) is used for both addition and concatenation of character of string.
- Here, binary operators (+) for addition, (-) for subtraction, (*) for multiplication, (/) for division, and (%) for modulus.
- Also unary operators are plus(+), negate(-), decrement(--), and increment(++).
- Increment(++) and Decrement(--) can be either prefix or postfix.
- Operator (**) is defined as exponentiation operator.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Arithmetic operators</title>
</head>
<body>
  <p id="demo"></p>
  <script type="text/javascript">
    function arithmeticOperation(){
      var a=5;
      var b=6;
      var c='9';
      var x=a+c;
      var z=a+b;
      var y=a**5;
      document.getElementById('demo').innerHTML="The sum is "+z+"<br>"+"The value of x
is "+x+"<br>The value of y is "+y;
    }
    arithmeticOperation();
  </script>
</body>
</html>
```

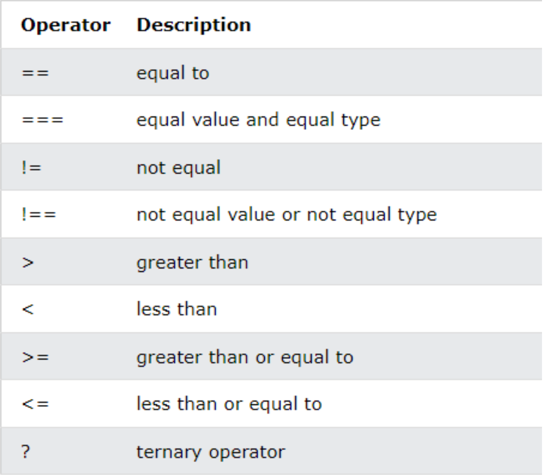
Note: Do other example yourself.

vii. **Assignment Operator**



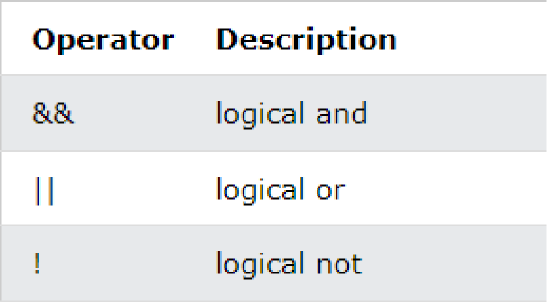
Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

viii. Comparision Operator



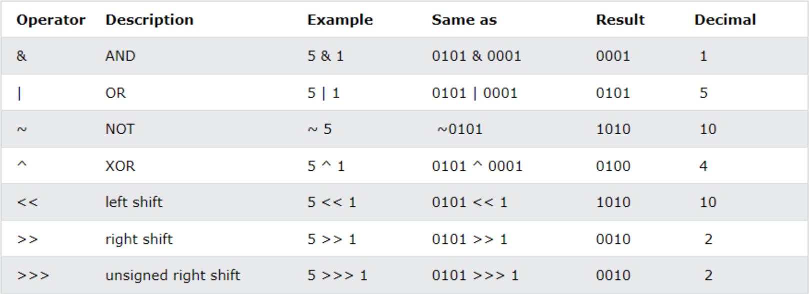
Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

ix. Logical Operator



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

x. Bitwise Operator



Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

xi. The typeof operator

The typeof operator returns the type of its single operand. This typeof operator is used to produces "number", "string", or "boolean" if the operand is of primitive type Number, String, or Boolean respectively. If the operand is an object or null, typeof produces “object”.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Arithmetic operators</title>
</head>
<body>
  <p id="demo"></p>
  <p id="demo1"></p>
  <p id="demo2"></p>
  <p id="demo3"></p>
  <script type="text/javascript">
    function Operation(){
      var a=5;
      var c=(++a)*3;
      var b=(a++)+"3";
      var car = {type:"Fiat", model:"500", color:"white"};
      document.getElementById('demo').innerHTML=typeof a;
      document.getElementById('demo1').innerHTML=typeof c;
      document.getElementById('demo2').innerHTML=typeof b;
      document.getElementById('demo3').innerHTML=typeof car;
    }
    Operation();
  </script>
</body>
</html>
```

xii. The *Date* object

- JavaScript facilitate about the current date and time in a program.
- In JavaScript , date and time are available through Date object and its rich collection of methods.
- A Date object is created with the new operator and the Date constructor, which has several forms.
- Some Date methods(Get methods and Set methods) are given within the example. So, understand yourself.
- Note, JavaScript counts month from 0 to 11. January is 0.December is 11.

Example:

```
 <!DOCTYPE html>
<html>
<head>
  <title>Date object</title>
</head>
<body>
  <p id="demo"></p>
  <p id="demo1"></p>
  <p id="demo2"></p>
  <p id="demo3"></p>
  <p id="demo4"></p>
  <p id="demo5"></p>
  <p id="demo6"></p>
  <p id="demo7"></p>
 <p id="demo8"></p>
 <script type="text/javascript">
  //creates a new date object with the current date and time
  var d = new Date();
  document.getElementById("demo").innerHTML = d;

  //getTime() method returns the number of milliseconds since January 1, 1970
  var da = new Date();
  document.getElementById("demo1").innerHTML = da.getTime();

  //getFullYear() method returns the year of a date as a four digit number
  var daa = new Date();
  document.getElementById("demo2").innerHTML= daa.getFullYear();

  //getMonth() method returns the month of a date as a number (0-11)
  var daaa = new Date();
  document.getElementById("demo3").innerHTML = daaa.getMonth() + 1;

  //gtDate() method returns the day of a date as a number (1-31)
  var daaaa = new Date();
  document.getElementById("demo4").innerHTML = daaaa.getDate();

  //getHours() method returns the hours of a date as a number (0-23)
  var daaaaa = new Date();
  document.getElementById("demo5").innerHTML = daaaaa.getHours();

  //getMinutes() method returns the minutes of a date as a number (0-59)
  var daaaaaa = new Date();
  document.getElementById("demo6").innerHTML = daaaaaa.getMinutes();

  //getSeconds() method returns the seconds of a date as a number (0-59)
  var daaaaaaa = new Date();
  document.getElementById("demo7").innerHTML = daaaaaaa.getSeconds();

  //getDay() method returns the weekday of a date as a number (0-6)
  var daaaaaaaa = new Date();
  document.getElementById("demo8").innerHTML = daaaaaaaa.getDay();
 </script>
</body>
</html>
```

Screen Output and Keyboard Input

- A JavaScript is interpreted when the browser finds the script or a reference to a separate script file in the body of the HTML document.
- The window in which the browser displays an HTML document is modeled with the *window* object.
- The window object includes two properties, *document* and *window*.
- The *document* property refers to the Document object.
- The *window* property is self-referential; it refers to the Window object.
- JavaScript can "display" data in different ways:
  - Writing into an HTML element, using *innerHTML*.
  - Writing into an HTML output using , *document.write()*.
  - Writing into an alert box, using *window.alert()*.

- Writing into the browser console, using *console.log()*.
- ***confirm*** method – opens a dialog window in which the method displays its string parameter, along with two buttons: *OK* and *Cancel*. Returns Boolean value true for *OK* and false for *Cancel*.
- ***prompt*** method – creates a dialog window that contains a text box used to collect a string of input from the user, which prompt returns as its value.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Screen output and keyboard input</title>
  <style type="text/css">
    #btn{
                border-radius: 5px;
                 height: 30px;
                width: 70px;
     }
     #btn:hover{
                 height: 50px;
                 width: 100px;
    }
  </style>
</head>
<body>
     <button onclick="myFunction()" id="btn">click</button>
    <p id="demo"></p>
    <button onclick="myFunction1()" id="btn">Prompt</button>
    <p id="demo1"></p>
  <script type="text/javascript">
    function myFunction(){
                var num1=45;
                var num2=89;
                 var sum=num1+num2;
      document.getElementById('demo').innerHTML="The sum of num1 and num2 is "+sum;
    }
                var num1=50;
                var num2=100;
                var sum=num1+num2;
                document.write(sum);
                function myFunction1(){
                var str1=prompt("Enter first number ","");
                var str2=prompt("Enter second number ","");
                var sum1=str1+str2;
                 window.alert("The sum is "+sum1);
    }
  </script> </body></html>
```

Function

- A JavaScript function is a block of reusable code designed to perform a particular task. This eliminates the need of writing the same code again and again.
- A JavaScript function is defined with the *function* keyword, followed by a **name**, followed by parentheses **()**.

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

Syntax:

```
function functionName(parameter1, parameter2,………)
{
        //code to be executed
}
```

Function Invocation:

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

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

Example 1:

```
<!DOCTYPE html>
<html>
<body>
        <h2>JavaScript Functions</h2>
        <p id="demo"></p>
        <p id="demo1"></p>
        <button onclick="btnClick()">change</button>
        <p id="demo2">The color of text will change if you click Change button</p>
<script>
        //function without argument
        function additionOfNumber(){
                var a = 20;
                var b = 40;
                var sum = a+b;
                console.log(`The sum of ${a} and ${b} is :${sum}`);
        //passing argument
        var x = myFunction(4, 3);
                document.getElementById("demo").innerHTML = x;
        //return value function with default value
        function myFunction(a=10, b) {
                return a * b;
        }
                document.getElementById("demo1").innerHTML =
                "The temperature is " + toCelsius(77) + " Celsius";
        function toCelsius(fahrenheit) {
                return (5/9) * (fahrenheit-32);
        }
        function btnClick(){
        document.getElementById('demo2').style.color="red";
        }
</script>
</body>
</html>
```

Anonymous Function:

An anonymous function is a function without a name.

Example:

```
<script type="text/javascript">
```

```
        //1. function without name , to call it, assign to variable called showMe.
        let showMe = function(){
                console.log("Do you know. I am anonymous function.");
        }
        showMe();

        //2. using as arguments
        setTimeout(function(){
                console.log("See you after 2 seconds.")
        },2000); //1second = 1000mili-seconds

        //3. Immediately invoked function execution
        (function(){
                console.log("called immediately after assigned.");
        })();

        //4. Immediately invoked with argument
        let animal = {
                name: 'dog',
                color: 'black',
        };
        (function(){
                console.log("I love "+animal.color+" colored " +animal.name+".");
        })(animal);

        //sum two number using anonymous function
        let addition = function(a,b){
                return a+b;
        }
        console.log("sum of:",addition(5,5));
    </script>
```

Arrow Function:

ES6 introduced arrow function expression that provides a shorthand for declaring anonymous functions:

Example: All Below example are shorthanded form of anonymous function using arrow function. Like example of anonymous function of above topic.

```
<script type="text/javascript">
        //1. function without name , to call it, assign to variable called showMe.
        let showMe = () =>console.log("Do you know. I am arrow function.");
        showMe();

        //used as arguments
        setTimeout(()=>
                console.log("See you after 2 seconds.")
        ,2000); //1second = 1000mili-seconds

        //Immediately invoked function execution
        (
                ()=>
                        console.log("called immediately after assigned.")
        )();

        //Immediately invoked with argument
        let animal = {
                name: 'dog',
                color: 'black',
        };
        (()=>{
                console.log("I love "+animal.color+" colored " +animal.name+".");
        })(animal);

        //sum two number using arrow function
        let addition = (a,b)=> a+b;
        console.log("sum of two number:",addition(5,5));
</script>
```

Array

- The **Array** object lets you store multiple values in a single variable.
- It stores a fixed-size sequential collection of elements of the same type.

  Syntax:

  var array_Name=[item1,item2,…..]; (good way to create)**OR**

  var array_Name=*new* Array(item1, item2,item3,….); (bad way to create)
- Array element can be access by referring to the **index** number.
- Array indexes start with 0. [0] is the first element.[1] is the second element.

Array Properties and Methods:

- ***length*** – property returns the length of an array(the number of array elements).
- ***toString()*** – method returns an array to string of(comma separated) array values.
- ***join()*** – method joins all array elements into a string. It behave like toString() method, but in addition you can specify separator.
- ***pop()*** – method removes last element of an array.(popping)
- ***push()*** – method adds an element to an array at end.(pushing)
- ***shift()*** – method removes first element and shift all other element to lower index.
- ***unshift()*** – method adds new element to an array at beginning.
- ***delete*** – operator is used to delete an elements. But it may leave undefined holes in the array. So prefer pop() or shift() methods.
- ***splice()*** – method can be used to add new element to an array, also returns an array with deleted elements.
- ***concat()*** - method creates a new array by merging(concatenating) existing arrays.
- ***slice()*** – method slices out a piece of an array into a new array.

Example:

```
<!DOCTYPE html>
<html>
<body>
    <h2>JavaScript Array Methods</h2>
    <p id="demo"></p>
    <p id="demo1"></p>
    <p id="demo2"></p>
    <p id="demo3"></p>
    <p id="demo4"></p>
    <p id="demo5"></p>
    <p id="demo6"></p>
    <p id="demo7"></p>
    <p id="demo8"></p>
```

```
        <p id="demo9"></p>
        <p id="demo10"></p>
<script>
        var fruits = ["Banana", "Orange", "Apple", "Mango"];//array
        var fruitsLen=fruits.length;   //length property
        document.getElementById("demo").innerHTML = "The length of array fruits is
        "+fruitsLen;
        document.getElementById("demo1").innerHTML = fruits.toString();
        document.getElementById("demo2").innerHTML = fruits.join("*");
        fruits.push("Kiwi");
        document.getElementById("demo3").innerHTML = fruits;
        document.getElementById("demo4").innerHTML = fruits.pop();
        document.getElementById("demo5").innerHTML = fruits;
        var vehicle=["car","bus","motorcycle","van"];
        document.getElementById('demo6').innerHTML=vehicle[0];
        document.getElementById("demo7").innerHTML = "Removed item is:
        "+"<b>"+vehicle.shift()+"</b>";
        document.getElementById("demo8").innerHTML = "Array after removing:
        "+"<b>"+vehicle+"</b>";
        document.getElementById("demo9").innerHTML = "Returns length of array:
        "+"<b>"+vehicle.unshift("tractor")+"</b>";
        document.getElementById("demo10").innerHTML = "Array after adding item:
        "+"<b>"+vehicle+"</b>";
</script> </body> </html>
```

Array Sorts:

- ***sort()*** – method sorts an array alphabetically. i.e. By default, the sort() function sorts values as **strings**. To sort numbers ***compare function*** is needed.
- ***reverse()*** – method reverses the elements in an array.
- ***Math.max.apply()*** – is used to find highest number in an array.
- ***Math.min.apply()*** – is used to find lowest number in an array.

Example:

```
<!DOCTYPE html>
<html>
<body>
        <h2>JavaScript Array Sort</h2>
        <button onclick="myFun()">Index</button>
        <p id="indexDemo">we can use index to access each element.</p>
        <button onclick="myFunction()">sort</button>
        <p id="demo"></p>
        <button onclick="myFunction1()">reverse</button>
        <p id="demo1"></p>
        <button onclick="myFunction2()">Number sort Ascending Order</button>
        <p id="demo2"></p>
        <button onclick="myFunction3()">Number sort Descending Order</button>
        <p id="demo3"></p>
        <p>The highest number is <span id="demo4"></span>.</p>
<script>
        var fruits = ["Banana", "Orange", "Apple", "Mango"];
        document.getElementById("demo").innerHTML = fruits;
        function myFun(){
       document.getElementById('indexDemo').innerHTML="The element at index 0 is "+fruits[0];
}
```

```
        function myFunction() {
         fruits.sort();
        document.getElementById("demo").innerHTML = fruits;
        }
        var vehicle=["car","bus","motorcycle","van"];
        document.getElementById("demo1").innerHTML = vehicle;
        function myFunction1(){
        // First sort the array
        vehicle.sort();
        // Then reverse it:
        vehicle.reverse();
        document.getElementById("demo1").innerHTML = vehicle;
        }
        var points = [40, 100, 1, 5, 25, 10];
        document.getElementById("demo2").innerHTML = points;
        //ascending order
        function myFunction2() {
        points.sort(function(a, b){return a - b}); //Compare function
        document.getElementById("demo2").innerHTML = points;
        }
        document.getElementById("demo3").innerHTML = points;
        function myFunction3() {
        points.sort(function(a, b){return b - a});
        document.getElementById("demo3").innerHTML = points;
        }
        document.getElementById("demo4").innerHTML = myArrayMax(points);
        function myArrayMax(arr) {
        return Math.max.apply(null, arr);
        }
</script>
</body>
</html>
```

Control Statements

Control Expressions

- The expressions upon which statement flow control can be based include primitive values, relational expressions, and compound expressions.
- The result of evaluating a control expression is one of the Boolean values true or false.

Operation	Operator
Is equal to	==
Is not equal to	!=
Is less than	<
Is greater than	>
Is less than or equal to	<=
Is greater than or equal to	>=
Is strictly equal to	===
Is strictly not equal to	!==

Conditional (if….else , if….elseif….else and switch) Statements

- Conditional statements are used to perform different actions based on different conditions.
- Use ***if*** to specify a block of code to be executed, if a specified condition is true.
- Use ***else*** to specify a block of code to be executed, if the same condition is false.
- Use elseif to specify a new condition to test, if the first condition is false.
- Use ***switch*** to specify many alternative blocks of code to be executed.

Example: if….elseif….else

```
<html>
<body>
         <p>Click the button to get a time-based greeting:</p>
        <button onclick="myFunction()">Try it</button>
       <p id="demo"></p>
<script>
         function myFunction() {
          var greeting;
          var time = new Date().getHours();
         if (time < 10) {
         greeting = "Good morning";
         } else if (time < 20) {
         greeting = "Good day";
         } else {
          greeting = "Good evening";
         }
         document.getElementById("demo").innerHTML = greeting;
         }
</script>
</body>
</html>
```

Example: Switch statement

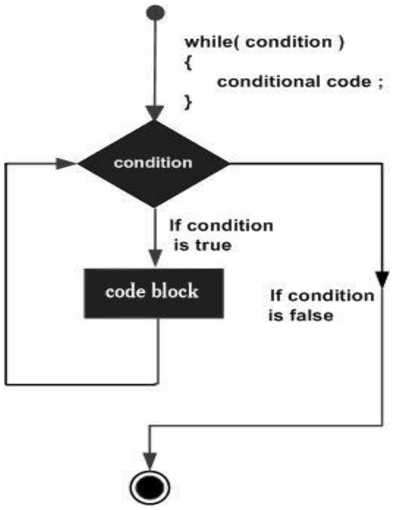
```
<!DOCTYPE html>
<html>
<body>
         <p id="demo"></p>
<script>
         var day;
        switch (new Date().getDay()) {
         case 0:
         day = "Sunday";
          break;
         case 1:
          day = "Monday";
         break;
         case 2:
         day = "Tuesday";
          break;
         case 3:
         day = "Wednesday";
          break;
```

```
                case 4:
                day = "Thursday";
                break;
                case 5:
                day = "Friday";
                break;
                case  6:
                day = "Saturday";
                }
               document.getElementById("demo").innerHTML = "Today is " + day;
    </script>
    </body>
    </html>
```

Loop Statement

While loop

Flow chart:



Syntax:

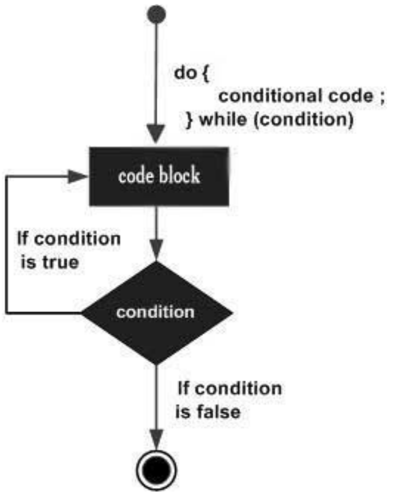
while(*expression*){
//statement(s) to be executed if expression is true
}

Example:

```
<html>
  <body>
    <script type="text/javascript">
        var count = 0;
        document.write("Starting Loop<br /> ");
         while (count < 10){
         document.write("Current Count : " + count + "<br />");
         count++;
        }
        document.write("Loop stopped!");
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

do…..while loop

Flow chart:



Syntax:

do{

//statement(s) to be executed

}while(*expression*)

Example:

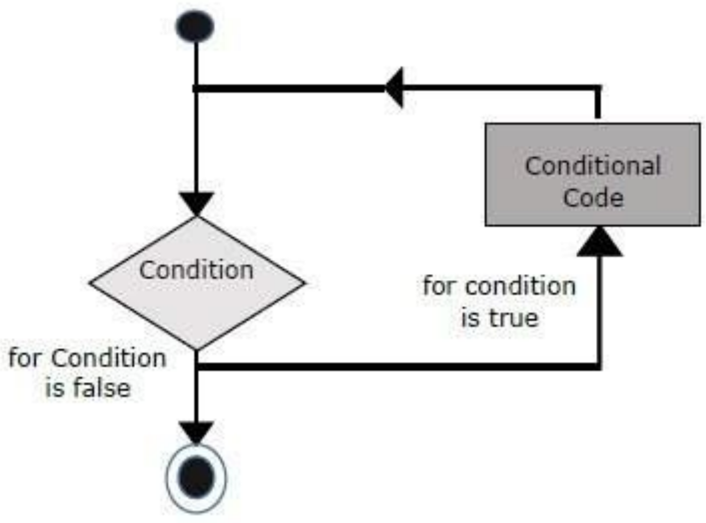
```
<html>
  <body>
    <script type = "text/javascript">
        var count = 0;
        document.write("Starting Loop" + "<br />");
        do {
          document.write("Current Count : " + count + "<br />");
          count++;
        }
        while (count < 5);
        document.write ("Loop stopped!");
    </script>
  </body>       </html>
```

for loop

It includes the following three important parts :

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

Flow chart:



Syntax:
for(initialization; test-condition; iteration-statement){
//statement(s) to be executed if expression is true
}

Example:

```
<html>
  <body>
    <script type = "text/javascript">
        var count;
        document.write("Starting Loop" + "<br />");
        for(count = 0; count < 10; count++) {
          document.write("Current Count : " + count );
          document.write("<br />");
        }
          document.write("Loop stopped!");
        </script>
      <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

for…in loop

The **for...in** loop is used to loop through an object's properties.

Syntax:
for(*variable_Name* **in** *object*){
//statement or block to be executed
}

Example:

```
<!DOCTYPE html>
<html>
<body>
      <h2>JavaScript Loops</h2>
      <p>The for/in statement loops through the properties of an object.</p>
      <p id="demo"></p>
<script>
      var txt = "";
      var person = {fname:"John", lname:"Doe", age:25};
```

```
                var x;
                for (x in person) {
                     txt += person[x] + " ";
            }
                document.getElementById("demo").innerHTML = txt;
            </script>
            </body>
            </html>
```

for…of loop:

The for… of loop is used to access array elements without appling coditions. Check for availability.

Example:

```
<script type="text/javascript">
        var fruits = ['apple','banana','grapes'];
        for(let x of fruits){
                console.log(x);
        }
</script>
```

forEach loop:

Uses callback function that iterate all elements exists in defined array. i.e. pass all elements to callback function parameter.

Example: callback function written in different form.

```
<script type="text/javascript">
        var animals = ['cat','dog','ox'];
        //using general function as callback
        console.log('Using general function.');
        animals.forEach(fun);
        function fun(value){
                console.log(value);
        }
        //using anonymous function as callback
        console.log('Using anonymous function.');
        animals.forEach(function(value){
                console.log(value);
        });
        //using arrow function as callback
        console.log('Using arrow function.');
        animals.forEach((value)=>console.log(value));
</script>
```

JavaScript Object Constructors and Prototype

- Sometimes we need a "blueprint" for creating many objects of the same "type".
- The way to create an "object type", is to use an object constructor function.
  Example:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
</head>
<body>
        <p id="demo"></p>
```

```
 <script>
        // Constructor function for Person objects
        function Person(first, last, age, eye) {
        this.firstName = first;
         this.lastName = last;
        this.age = age;
        this.eyeColor = eye;
         }
        // Create two Person objects
        var myFather = new Person("John", "Doe", 50, "blue");
        var myMother = new Person("Sally", "Rally", 48, "green");
        // Display age
        document.getElementById("demo").innerHTML = "My father is " + myFather.age + ". My
        mother is " + myMother.age + ".";
</script>
</body> </html>
```

this keyword

- In a function definition, this refers to the "owner" of the function.
- The JavaScript *this* keyword refers to the object it belongs to.
- It has different values depending on where it is used.
  - In a method, ***this*** refers to the **owner object**.
  - Alone, ***this*** refers to the **global object**.
  - In a function, ***this*** refers to the **global object**.
  - In a function, in strict mode, ***this*** is ***undefined***.
  - In an event, ***this*** refers to the **element** that received the event.
  - Methods like **call()** and **apply()** can refer ***this*** to **any object**.

Example:

```
<html>
<body>
    <h2>The JavaScript <b>this</b> Keyword</h2>
    <p>In this example, <b>this</b> represents the <b>person</b> object.</p>
    <p>Because the person object "owns" the fullName method.</p>
    <p id="demo"></p>
    <p>In this example <strong>this</strong> refers to person2, even if it is a method of person1:</p>
    <p id="demo1"></p>
    <button onclick="this.style.display='none'">Click to Remove Me!</button>
<script>
    // Create an object:
    var person = {
    firstName: "John",
    lastName : "Doe",
     id     : 5566,
     fullName : function() {
    return this.firstName + " " + this.lastName;
    }
    };
    // Display data from the object(person):
    document.getElementById("demo").innerHTML = person.fullName();

    //using method call()
    var person1 = {
    fullName: function() {
    return this.firstName + " " + this.lastName;
     }
    }
    var person2 = {
    firstName:"Ram",
    lastName: "Prasad",
    }
    var x = person1.fullName.call(person2);
    document.getElementById("demo1").innerHTML = x;
</script>
</body></html>
```

JavaScript HTML DOM

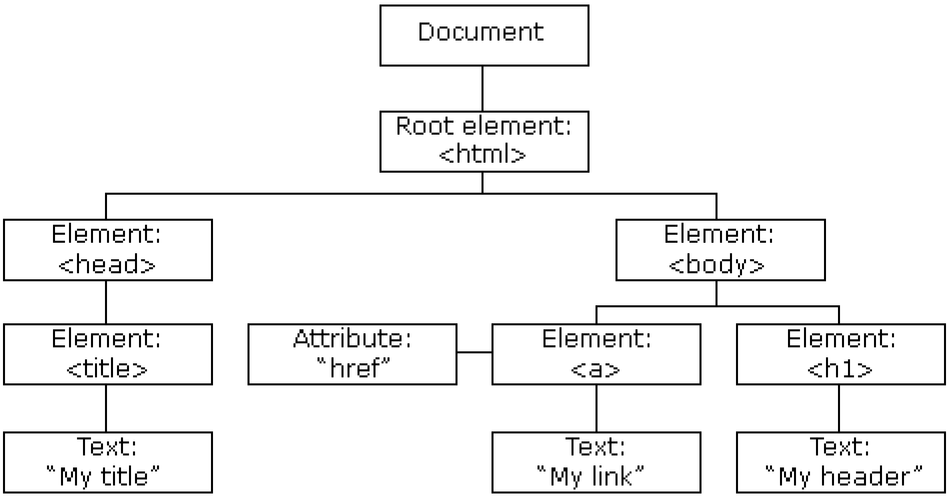
- DOM stands for Document Object Model has been under development by the W3C since the mid-1990s. DOM 3 is the current approved version.
- The DOM defines a standard for accessing documents.
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The original motivation for the standard DOM was to provide a specification that would allow Java programs and JavaScript scripts that deal with HTML documents to be portable among various browsers.
- The DOM is an application programming interface(API) that defines an interface between HTML documents and application programs.
- The W3C DOM standard is separated into 3 different parts:
    - Core DOM - standard model for all document types
    - XML DOM - standard model for XML documents
    - HTML DOM - standard model for HTML documents

  But here we study only about HTML DOM in this chapter.

  The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
    - The HTML elements as **objects**
    - The **properties** of all HTML elements
    - The **methods** to access all HTML elements
    - The **events** for all HTML elements

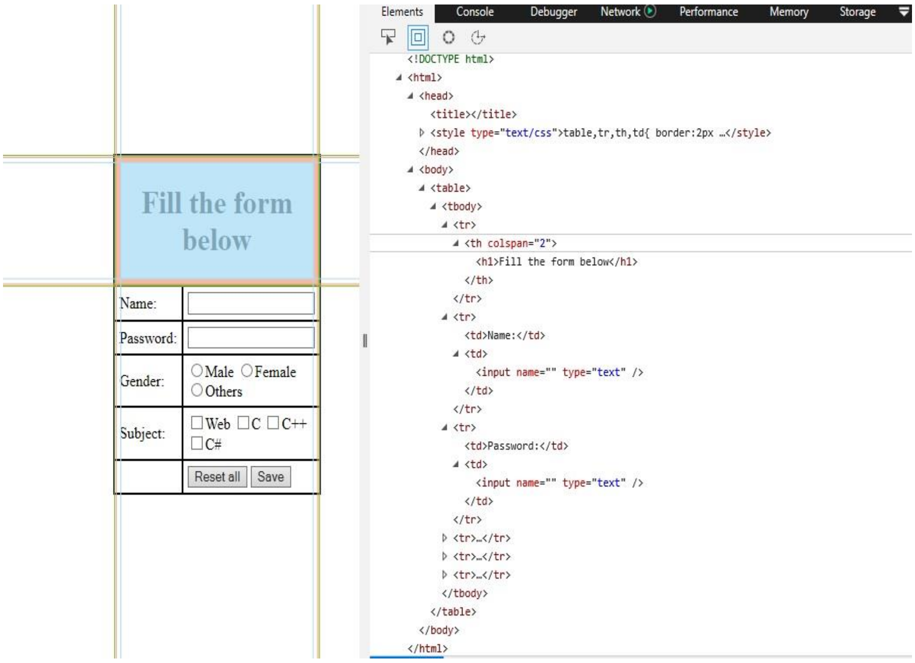
  In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

  The **HTML DOM** model is constructed as a tree of **Objects**:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page



```
<!DOCTYPE html>
◢ <html>
  ◢ <head>
      <title></title>
    ▷ <style type="text/css">table,tr,th,td{ border:2px …</style>
    </head>
  ◢ <body>
    ◢ <table>
      ◢ <tbody>
        ◢ <tr>
          ◢ <th colspan="2">
              <h1>Fill the form below</h1>
            </th>
          </tr>
        ◢ <tr>
            <td>Name:</td>
          ◢ <td>
              <input name="" type="text" />
            </td>
          </tr>
        ◢ <tr>
            <td>Password:</td>
          ◢ <td>
              <input name="" type="text" />
            </td>
          </tr>
        ▷ <tr>…</tr>
        ▷ <tr>…</tr>
        ▷ <tr>…</tr>
        </tbody>
      </table>
    </body>
</html>
```

figure: The DOM structure of table.html with IE9

JavaScript HTML DOM Document

As you know document object represents your web page and HTML DOM ***document*** object is the owner of all other objects . So document object is used to access any element in and HTML page. Some examples to access and manipulates HTML that uses the ***document*** object ,given below.

Finding HTML Elements:

If you want to manipulate HTML elements than find elements first and access them to JavaScript. There are several ways :

i. **Finding HTML Element by Id:** Easiest way to find an HTML element in the DOM. To get elements using id use *document.getElementById(‘idName’)* .

Example:

```
<!DOCTYPE html>
<html>
<head>
        <title>Finding Elements by Id</title>
</head>
<body>
        <form>
                First Number:
                <input type="Number" name="fnum" id="f-num" value="5">
                </br></br>
                Second Number:
                <input type="Number" name="snum" id="s-num" value="8">
                </br></br>
                <input type="button" name="btn" onclick="add()" value="Add">
        </form>
        <p>Sum value: <span id="result"></span></p>

        <script type="text/javascript">
                function add(){
                //get first number and assign to firstNumber
                        var firstNumber = document.getElementById('f-num').value;
                //typecasting ,String to Number
                        firstNumber = Number(firstNumber);
                //get second number and assign to secondNumber
                        var secondNumber = document.getElementById('s-num').value;
                //typecasting, String to Number
                        secondNumber = Number(secondNumber);
                //sum operation (firstNumber and secondNumber)
                        var sum = firstNumber + secondNumber;
                //get span element using its id and display output
                        document.getElementById('result').innerHTML = sum;
                }
        </script>
</body>
</html>
```

ii. **Finding HTML Element by Tag name:** To find any elements from document use *document.getElementsByTagName(‘tagName’).*

Example:

```
<!DOCTYPE html>
<html>
<head>
	<title>Finding Elements by TagName.</title>
</head>
<body>
	<form>
		First Number:
		<input type="Number" name="fnum" value="5">
		</br></br>
		Second Number:
		<input type="Number" name="snum" value="8">
		</br></br>
		<input type="button" name="btn" onclick="add()" value="Add">
	</form>
	<p>Sum value: <span id="result"></span></p>

	<script type="text/javascript">
		function add(){
		//get input tags from document
			var inputTag = document.getElementsByTagName('input');
		//get first number and assign to firstNumber
			var firstNumber = inputTag[0].value;
		//typecasting ,String to Number
			firstNumber = Number(firstNumber);
		//get second number and assign to secondNumber
			var secondNumber = inputTag[1].value;
		//typecasting, String to Number
			secondNumber = Number(secondNumber);
		//sum operation (firstNumber and secondNumber)
			var sum = firstNumber + secondNumber;
		//get span tags and display output from document
			var spanTag = document.getElementsByTagName('span');
			spanTag[0].innerHTML = sum;
		}
	</script>
</body>
</html>
```

iii. **Finding HTML Elements by class name:** To find any elements from document use *document.getElementsByClassName('className')*.

Example:

```
<!DOCTYPE html>
<html>
<head>
	<title>Finding Elements By ClassName.</title>
</head>
<body>
	<form>
		First Number:
		<input type="Number" name="fnum" class="num" value="5">
```

```
        </br></br>
        Second Number:
        <input type="Number" name="snum" class="num" value="8">
        </br></br>
        <input type="button" name="btn" onclick="add()" value="Add">
    </form>
    <p>Sum value: <span class="num"></span></p>
    <script type="text/javascript">
        function add(){
        //get elements using class name from document
            var usingClassName = document.getElementsByClassName('num');
        //get first number and assign to firstNumber
            var firstNumber = usingClassName[0].value;
        //typecasting ,String to Number
            firstNumber = Number(firstNumber);
        //get second number and assign to secondNumber
            var secondNumber = usingClassName[1].value;
        //typecasting, String to Number
            secondNumber = Number(secondNumber);
        //sum operation (firstNumber and secondNumber)
            var sum = firstNumber + secondNumber;
        //get span element using class name and display output from document
            usingClassName[2].innerHTML = sum;
        }
    </script>
</body>
</html>
```

iv. **Finding HTML Elements by CSS selectors:** *querySelectorAll()* is used to find all elements that match a specified CSS selectors(id,class name, types, attributes, tag, value of attributes etc).

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Finding Elements By querySelectorAll.</title>
</head>
<body>
    <form>
        First Number:
        <input type="Number" name="fnum" class="num" value="5">
        </br></br>
        Second Number:
        <input type="Number" name="snum" class="num" value="8">
        </br></br>
        <input type="button" name="btn" onclick="add()" value="Add">
    </form>
    <p>Sum value: <span class="num"></span></p>

    <script type="text/javascript">
        function add(){
        //get elements using class name from document
```

```
            var usingQuerySelector = document.querySelectorAll('.num');
        //get first number
            var firstNumber = usingQuerySelector[0].value;
        //typecasting to number
            firstNumber = Number(firstNumber);
        //get second number
            var secondNumber = usingQuerySelector[1].value;
        //typecasting to number
            secondNumber = Number(secondNumber);
        //display sum
            var sum = firstNumber + secondNumber;
            usingQuerySelector[2].innerHTML = sum;
        }
    </script>
</body>
</html>
```

v. **Finding HTML Elements by HTML object collections:** There are more HTML object collections (***Example***: *document.body, document.anchors, document.images, document.links, document.documentElement, document.title* , etc).

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Finding Elements By ClassName.</title>
</head>
<body>
    <form id="additionForm">
        First Number:
        <input type="Number" name="fnum" value="5">
        </br></br>
        Second Number:
        <input type="Number" name="snum" value="8">
        </br></br>
        <input type="button" name="btn" onclick="add()" value="Add">
    </form>
    <p>Sum value: <span id="result"></span></p>

    <script type="text/javascript">
        function add(){
        /*form is a collection of inputs , button ,...
            i.e. collection object.
            So to get form elements use form id
                and forms keyword for form.
        */
            var form_data = document.forms['additionForm'];
            console.log(form_data); //output form in console
        //get first number
            var firstNumber = form_data.elements[0].value;
            console.log(firstNumber); //display firstNumber
            console.log(typeof(firstNumber)); //check type of firstNumber
            firstNumber = Number(firstNumber); //typecasting
```

```
                    console.log(typeof(firstNumber)); //check type of firstNumber
            //get second number
                    var secondNumber = form_data.elements[1].value;
                    secondNumber = Number(secondNumber);
            //display output On browser i.e through HTML document
                    var sum = firstNumber + secondNumber;
                    document.getElementById('result').innerHTML = sum;
            }
        </script>
</body>
</html>
```

Adding and Deleting Elements:

document.createElement(element) – create an HTML element
document.removeChild(element) – remove an HTML element
document.appendChild(element) – add an HTML element
document.replaceChild(element) – replace an HTML element

Event and Event Handling

- **Event**s are action that are detected by JavaScript.
- JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
- To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute.
- Examples of HTML events:
  - When a user clicks the mouse
  - When a web page has loaded
  - When an image has been loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted
  - When a user strokes a key
- When an event occur then this action get response from script , this process is called **event handling**.

  In following example, the content of the ***<h1>*** element is changed when a user clicks on it:

```
<!DOCTYPE html>
<html>
<body>
   <h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>
</body>
</html>
```

Some example of JavaScript event type are given below:

onclick event type:

This is the most frequently used event type which occurs when a user clicks the left button of his mouse.

Example:

```
<html>
  <head>
    <title>JS event </title>
  </head>
  <body>
    <p>Click the following button and see result</p>
    <button id="myBtn">Try it</button>
    <p id="demo"></p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
    </form>
    <script type = "text/javascript">
                function sayHello() {
                alert("Hello World")
                }
                document.getElementById("myBtn").onclick = displayDate;
                function displayDate() {
                document.getElementById("demo").innerHTML = Date();
                }
    </script>
  </body>
</html>
```

onsubmit event type:

onsubmit is an event that occurs when you try to submit a form.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
<form method="post" action="test9.php" target="_blank">
    <br><input type="text" name="firstName">
    <br><input type="text" name="secondName">
    <br><input type="submit" name="submit">
</form>
</body>
</html>
```

Note: When submit button is clicked ; form will be submitted and have action on php script named *"test9.php"* i.e. given below:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
    Welcome <?php echo $_POST["firstName"]; ?><br>
    Your address is <?php echo $_POST["address"];?>
    <br>Hello <?php echo $_POST["firstName"]; ?>
</body>
</html>
```

onmouseover and onmouseout event type:

The onmouseover event triggers when you bring your mouse over any element and the onmouseout triggers when you move your mouse out from that element.

Example:

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

<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>
<script>
function mOver(obj) {
  obj.innerHTML = "Release Me";
}
function mOut(obj) {
  obj.innerHTML = " Thank You ";
}
</script>
</body>
</html>
```

onmousedown and onmouseup event type:

The ***onmousedown***, ***onmouseup***, and ***onclick*** events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

Example:

```
<!DOCTYPE html>
<html>
<body>
    <div onmousedown="mDown(this)" onmouseup="mUp(this)"
    style="background-color:#D94A38;width:90px;height:20px;padding:40px;">
    Click Me</div>
<script>
    function mDown(obj) {
    obj.style.backgroundColor = "#1ec5e5";
    obj.innerHTML = "Release Me";
}
    function mUp(obj) {
    obj.style.backgroundColor="#D94A38";
    obj.innerHTML="Thank You";
}
</script>
</body>
</html>
```

onchange event type:

The onchange event is often used in combination with validation of input fields.

Example:

```
<html>
<head>
<script>
```

```
    function myFunction() {
    var x = document.getElementById("fname");
    x.value = x.value.toUpperCase();
}
</script>
</head>
<body>
    Enter your name: <input type="text" id="fname" onchange="myFunction()">
    <p>When you leave the input field,  the input text transfer to upper case.</p>
</body>
</html>
```

onblur and onfocus event type:

The ***onblur*** event occurs when an object loses focus. The onblur event is most often used with form validation code (e.g. when the user leaves a form field).

The ***onfocus*** event occurs when an element gets focus. The onfocus event is most often used with <input>, <select>, and <a>.

Example:

```
<!DOCTYPE html>
<html>
<body>
    <p>This example uses the HTML DOM to assign an "onfocus" event to an input element.</p>
    <!-- When click for input it get focus but when click out the input field it looses focus i.e. blur.
    -->
    Enter your name: <input type="text" id="fname">
<script>
    document.getElementById("fname").onfocus = function() {myFunction()};
    function myFunction() {
            document.getElementById("fname").style.backgroundColor = "red";
    }

    document.getElementById("fname").onblur = function() {myFunction1()};
    function myFunction1() {
            document.getElementById("fname").style.backgroundColor = "yellow";
    }
</script>
</body>
</html>
```

There are a lot of event types that are used by JS . Some of them are listed below:

Attribute	Value	Description
Offline	script	Triggers when the document goes offline
Onabort	script	Triggers on an abort event
onafterprint	script	Triggers after the document is printed
onbeforeonload	script	Triggers before the document loads
onbeforeprint	script	Triggers before the document is printed

oncanplay	script	Triggers when media can start play, but might has to stop for buffering
oncanplaythrough	script	Triggers when media can be played to the end, without stopping for buffering
onchange	script	Triggers when an element changes
oncontextmenu	script	Triggers when a context menu is triggered
ondblclick	script	Triggers on a mouse double-click
ondrag	script	Triggers when an element is dragged
ondragend	script	Triggers at the end of a drag operation
ondragenter	script	Triggers when an element has been dragged to a valid drop target
ondragleave	script	Triggers when an element is being dragged over a valid drop target
ondragover	script	Triggers at the start of a drag operation
ondragstart	script	Triggers at the start of a drag operation
ondrop	script	Triggers when dragged element is being dropped
ondurationchange	script	Triggers when the length of the media is changed
onemptied	script	Triggers when a media resource element suddenly becomes empty.
onended	script	Triggers when media has reach the end
onerror	script	Triggers when an error occur
onformchange	script	Triggers when a form changes
onforminput	script	Triggers when a form gets user input
onhaschange	script	Triggers when the document has change
oninput	script	Triggers when an element gets user input
oninvalid	script	Triggers when an element is invalid
onkeydown	script	Triggers when a key is pressed
onkeypress	script	Triggers when a key is pressed and released
onkeyup	script	Triggers when a key is released
onload	script	Triggers when the document loads
onloadeddata	script	Triggers when media data is loaded
onloadedmetadata	script	Triggers when the duration and other media data of a media element is loaded
onloadstart	script	Triggers when the browser starts to load the media data
onmessage	script	Triggers when the message is triggered
onmousemove	script	Triggers when the mouse pointer moves

onmousewheel	script	Triggers when the mouse wheel is being rotated
onoffline	script	Triggers when the document goes offline
onoine	script	Triggers when the document comes online
ononline	script	Triggers when the document comes online
onpagehide	script	Triggers when the window is hidden
onpageshow	script	Triggers when the window becomes visible
onpause	script	Triggers when media data is paused
onplay	script	Triggers when media data is going to start playing
onplaying	script	Triggers when media data has start playing
onpopstate	script	Triggers when the window's history changes
onprogress	script	Triggers when the browser is fetching the media data
onratechange	script	Triggers when the media data's playing rate has changed
onreadystatechange	script	Triggers when the ready-state changes
onredo	script	Triggers when the document performs a redo
onresize	script	Triggers when the window is resized
onscroll	script	Triggers when an element's scrollbar is being scrolled
onseeked	script	Triggers when a media element's seeking attribute is no longer true, and the seeking has ended
onseeking	script	Triggers when a media element's seeking attribute is true, and the seeking has begun
onselect	script	Triggers when an element is selected
onstalled	script	Triggers when there is an error in fetching media data
onstorage	script	Triggers when a document loads
onsuspend	script	Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched
ontimeupdate	script	Triggers when media changes its playing position
onundo	script	Triggers when a document performs an undo
onunload	script	Triggers when the user leaves the document
onvolumechange	script	Triggers when media changes the volume, also when volume is set to "mute"
onwaiting	script	Triggers when media has stopped playing, but is expected to resume

Moving Elements and Element Positioning

We can move HTML elements in JavaScript as follows:

```
<html>
<head>
<script>
function moveDiv(obj){
  if(obj.align=="center"){
    obj.align="right";
  }
  else{
    obj.align="center";
  }
}
</script>
</head>
<body>
   <div id="myDiv" onclick="moveDiv(this)" align="center">
   Hello, World        </div>
</body>
</html>
```

The position property sets or returns the type of positioning method used for an element (static, relative, absolute or fixed).

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
   #myDIV {
    border: 1px solid black;
    background-color: lightblue;
    width: 300px;
   height: 300px;
   position: relative;
    top: 20px;
   }
</style>
</head>
<body>
   <p>Click the "Try it" button to change the position property of the DIV element:</p>
   <button onclick="myFunction()">Try it</button>

   <div id="myDIV">
               <p>This DIV element is placed 20 pixels from the top of its original position.</p>
               <p>Click the button to set the position property to "absolute" and see what
   happens.</p>
               <p>It will then be placed 20 pixels from the top the page.</p>
   </div>
<script>
               function myFunction() {
                document.getElementById("myDIV").style.position = "absolute";
}
</script>
</body>
```

Changing colors and fonts

Using JavaScript you can change any properties of CSS for any elements of HTML. The following example is for changing font color and its size.

Example:

```
<!DOCTYPE html>
<html>
<head>
            <title>Changing Color and Font size of content</title>
  <script type="text/javascript">
            function myFunction(){
            var x=document.getElementById('demo');
            x.style.color="Red";
            x.style.fontSize="50px";
            x.style.fontWeight="bold";
            x.style.fontStyle="italic";
    }
  </script>
</head>
<body>
            <p id="demo">Change me please!</p>
            <button onclick="myFunction()">Change</button>
</body>
</html>
```

Dynamic content and stacking elements

- The z-index attribute determines which element is in front and which are covered by the front element
- The JavaScript property associated with the z-index attribute is zIndex
- z-index can be changed dynamically (by changing zIndex)
- An image element can have an onclick attribute, so images can be clicked to trigger event handlers
- Anchors can also trigger event handlers when they are clicked

  Example:

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <title> Dynamic stacking of images </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript">

            function toTop(obj){
             var topp = "airplane1";
            var domTop = document.getElementById(topp).style;
            var domNew = document.getElementById(obj).style;
            domTop.zIndex = "0";
            domNew.zIndex = "10";
            topp = obj;
  }
    </script>
    <style type = "text/css">
            .plane1 {
            position: absolute;
            top: 0;
            width: 300px;
            height: 400px;
            left: 0;
            z-index: 0;
}
            .plane2 {
            position: relative;
            top: 50px;
            left: 110px;
            z-index: 0;}
            .plane3 {
            position: absolute;
            top: 100px;
             left: 220px;
            z-index: 0;}
   </style>
  </head>
  <body>
            <p>
            <img class ="plane1"  id = "airplane1" src = "bas2.jpg" alt = "(Picture of an          airplane)"
onclick = "toTop('airplane1')" />
            <img class ="plane2"  id = "airplane2" src = "karnali.jpg" alt = "(Picture of an        airplane)"
onclick = "toTop('airplane2')" />
            <img class ="plane3"  id = "airplane3" src = "bas2.jpg" alt = "(Picture of an          airplane)"
onclick = "toTop('airplane3')" />
            </p>
  </body>
</html>
```

Locating the mouse cursor and reacting to a mouse click

- The *clientX* property returns the horizontal coordinate (according to the client area) of the mouse pointer when a mouse event was triggered.(similarly we can use *clientY* for vertical). In current window.
- The *screenY* property returns the vertical coordinate (according to the users computer screen) of the mouse pointer when an event was triggered(similarly we can use *screenX* for horizontal coordinates).

Example:

```
<!DOCTYPE html>
<html>
<head>
            <style>
            div {
            width: 500px;
            height: 300px;
            border: 1px solid black;
            text-align: center;
            }
</style>
</head>
<body>
            <div onclick="showCoords(event)"><p id="demo"> </p></div>
            <p><strong>Tip:</strong> Try to click different places in the div.</p>
<script>
            function showCoords(event) {
             var cX = event.clientX;
             var sX = event.screenX;
```

```
                 var cY = event.clientY;
                var sY = event.screenY;
                var coords1 = "client - X: " + cX + ", Y coords: " + cY;
                var coords2 = "screen - X: " + sX + ", Y coords: " + sY;
                document.getElementById("demo").innerHTML = coords1 + "<br>" + coords2;
                }
</script>
</body>
</html>
```

Dragging and dropping elements

Drag and drop elements allows us for dragging and dropping elements from one position to another.

Steps:

1. First make an element ***draggable*** , set ***draggable*** attribute to ***true***.
2. Then specify , what should happen when element is dragged. ***ondragstart*** attribute calls a function **(*drag(event)*),** that specifies what data to be dragged and the ***dataTransfer.setData()*** method sets the data type and the value of the dragged data.
3. Then also specify where to drop dragged data (i.e. specified by ***ondragover*** event). Also to allow a ***drop*** , we must prevent the default handling of the element.
4. ***preventDefault()*** method is used to prevent the browser default handling of the data.
5. When the dragged data is dropped , a drop event occurs. The ***ondrop*** attribute calls a function , ***drop(event).***
6. Append the dragged element into the drop element.

Example:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
    #div1 {
                 width: 350px;
                height: 71px;
                 padding: 10px;
                 border: 1px solid #aaaaaa;
                }
</style>
<script>
                function allowDrop(ev) {
                 ev.preventDefault();
                }
                function drag(ev) {
                 ev.dataTransfer.setData("text", ev.target.id);
                }
                function drop(ev) {
                ev.preventDefault();
                var data = ev.dataTransfer.getData("text");
                ev.target.appendChild(document.getElementById(data));
                }
```

```
</script>
</head>
<body>
        <p>Drag the  image into the rectangle:</p>
        <div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
        <br>
        <img id="drag1" src="karnali.jpg" draggable="true" ondragstart="drag(event)"
width="336" height="70">
</body>
</html>
```

Display and Visibility in JavaScript

The **display** property sets or returns the element's display type. The **visibility** property sets or returns whether an element should be visible. Both properties allows author to show or hide an element. However, if you set ***display: none***, it hides the entire element, while ***visibility: hidden*** means that the contents of the element will be invisible, but the element stays in its original position and size.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Display and Visibility Properties</title>
</head>
<body>
  <p id="hideShow"> click HIDE button to hide and SHOW button to show</p>
  <button onclick="hideDisplay()">Hide</button>
  <button onclick="showDisplay()">SHOW</button>
  <p id="hideShow1"> click HIDDEN button to hide and VISIBLE button to show</p>
  <button onclick="hideVisibility()">HIDDEN</button>
  <button onclick="showVisibility()">VISIBLE</button>
 <script type="text/javascript">
          var disp=document.getElementById('hideShow').style;
          function hideDisplay(){
          disp.display="none";
           }
           function showDisplay(){
           disp.display="block";
          }

          var visi=document.getElementById('hideShow1').style;
          function hideVisibility(){
           visi.visibility="hidden";
           }
          function showVisibility(){
           visi.visibility="visible";
           }
 </script>
</body>
</html>
```

Regular Expression(Pattern Matching)

A regular expression is an object that describes a pattern of characters. The JavaScript **RegExp** class represents regular expressions, and both String and **RegExp** define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

Syntax:

var pattern=new RegExp(pattern, attributes);
or simply
var pattern=/pattern/attributes;

Here is the description of the parameters −

- **pattern** − A string that specifies the pattern of the regular expression or another regular expression.
- **Attributes/modifiers** − An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multi-line matches, respectively.

Brackets[]:

Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Sr.No.	Expression & Description
1	**[...]** Any one character between the brackets.
2	**[^...]** Any one character not between the brackets.
3	**[0-9]** It matches any decimal digit from 0 through 9.
4	**[a-z]**It matches any character from lowercase **a** through lowercase **z**.
5	**[A-Z]**It matches any character from uppercase **A** through uppercase **Z**.
6	**[a-Z]**It matches any character from lowercase **a** through uppercase **Z**.

Quantifiers:

Sr.No.	Expression & Description
1	**p+** It matches any string containing one or more p's.
2	**p*** It matches any string containing zero or more p's.
3	**p?** It matches any string containing at most one p.
4	**p{N}** It matches any string containing a sequence of **N** p's

5	**p{2,3}** It matches any string containing a sequence of two or three p's.
6	**p{2, }** It matches any string containing a sequence of at least two p's.
7	**p$** It matches any string with p at the end of it.
8	**^p** It matches any string with p at the beginning of it.

Examples:

Following examples explain more about matching characters.

Sr.No.	Expression & Description
1	**[^a-zA-Z]** It matches any string not containing any of the characters ranging from **a** through **z** and **A** through Z.
2	**p.p** It matches any string containing **p,** followed by any character, in turn followed by another **p**.
3	**^.{2}$** It matches any string containing exactly two characters.
4	**<b>(.*)</b>** It matches any string enclosed within <b> and </b>.
5	**p(hp)*** It matches any string containing a **p** followed by zero or more instances of the sequence **hp**.

Literal Characters:

Sr.No.	Character & Description
1	Alphanumeric Itself
2	**\0** The NUL character (\u0000)
3	**\t** Tab (\u0009
4	**\n** Newline (\u000A)
5	**\v** Vertical tab (\u000B)
6	**\f** Form feed (\u000C)
7	**\r** Carriage return (\u000D)

8	**\xnn** The Latin character specified by the hexadecimal number nn; for example, \x0A is the same as \n
9	**\uxxxx** The Unicode character specified by the hexadecimal number xxxx; for example, \u0009 is the same as \t
10	**\cX** The control character ^X; for example, \cJ is equivalent to the newline character \n

Metacharacters:

A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.

Sr.No.	Character & Description
1	**.** a single character
2	**\s** a whitespace character (space, tab, newline)
3	**\S** non-whitespace character
4	**\d** a digit (0-9)
5	**\D** a non-digit
6	**\w** a word character (a-z, A-Z, 0-9, _)
7	**\W** a non-word character
8	**[\b]** a literal backspace (special case).
9	**[aeiou]** matches a single character in the given set
10	**[^aeiou]** matches a single character outside the given set
11	**(foo|bar|baz)** matches any of the alternatives specified

Modifiers:

Several modifiers are available that can simplify the way you work with **regexps,** like case sensitivity, searching in multiple lines, etc.

Sr.No.	Modifier & Description
1	**i** Perform case-insensitive matching.
2	**m** Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary
3	**g** Performs a global match that is, find all matches rather than stopping after the first match.

Example: **global** is a read-only boolean property of RegExp objects. It specifies whether a particular regular expression performs global matching, i.e., whether it was created with the "g" attribute. Returns "TRUE" if the "g" modifier is set, "FALSE" otherwise.

```
<html>
  <head>
    <title>JavaScript RegExp global Property</title>
  </head>
  <body>
    <script type = "text/javascript">
      var re = new RegExp( "string" );
      if ( re.global ) {
        document.write("Test1 - Global property is set");
      }
      else {
        document.write("Test1 - Global property is not set");
      }
      re = new RegExp( "string", "g" );
      if ( re.global ) {
        document.write("<br />Test2 - Global property is set");
      }
      else {
        document.write("<br />Test2 - Global property is not set");
      }
    </script>
  </body>
</html>
```

test() method: It searches a string for a pattern, and returns true or false, depending on the result.

Example 1:

```
<!DOCTYPE html>
<html>
<body>
    <h2>JavaScript Regular Expressions</h2>
    <p>Search for an "e" in the next paragraph:</p>
    <p id="p01">The best things in life are free!</p>
    <p id="demo"></p>
<script>
    text = document.getElementById("p01").innerHTML;
    document.getElementById("demo").innerHTML = /e/.test(text);

    let firstExp = /namaste/gi;//creating reqular expression
    let greeting = "Namaste sir. You didn\'t response my namaste.";
    //test()
                let result = firstExp.test(greeting);
                console.log(result);
                console.log(greeting.replace(firstExp,'Hello'));
</script>
</body>
</html>
```

exec() Method: It searches a string for a specified pattern, and returns the found text as an object. If no match is found, it returns an empty *(null)* object.

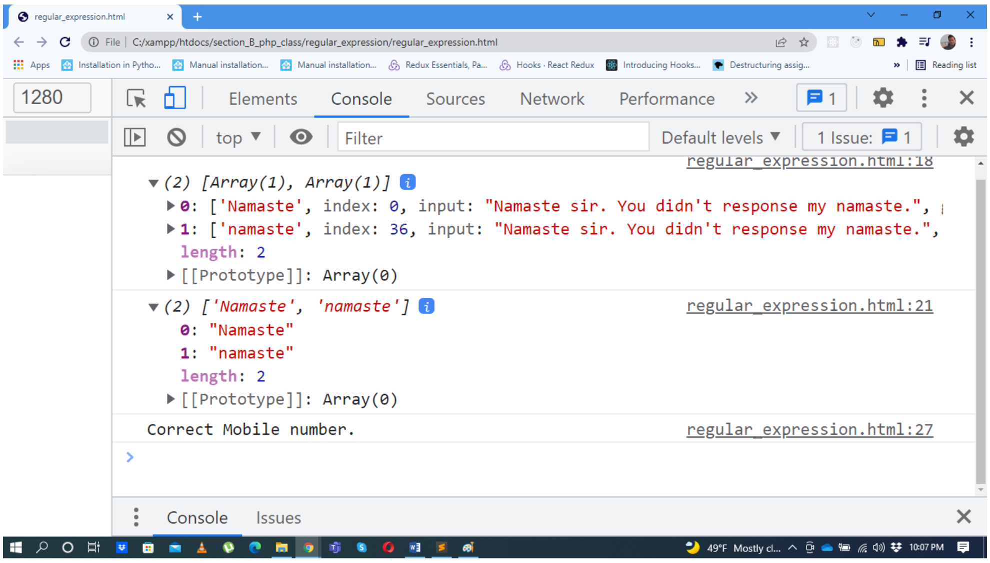
Example 2:

```
<!DOCTYPE html>
<html>
<body>
    <h2>JavaScript Regular Expressions</h2>
    <p id="demo"></p>
    <script>
    var obj = /t/i.exec("The best things in life are free!");
    document.getElementById("demo").innerHTML =
    "Found " + obj[0] + " in position " + obj.index + " in the text: " + obj.input;
</script>
</body>
</html>
```

Example 3:

```
<script type="text/javascript">
    let firstExp = /namaste/gi;//creating reqular expression
    let greeting = "Namaste sir. You didn\'t response my namaste.";
    //exec()
            let matchedPattern = [];
            let matchValue;
            do{
                    matchValue = firstExp.exec(greeting);
                    if(matchValue){
                            matchedPattern.push(matchValue);
                    }
            }while(matchValue !=null);
            console.log(matchedPattern);

    //match()
            console.log(greeting.match(firstExp));

            //mobile number validation
            let phoneNumber = '9898888888';
            let regPhone = /^\98[0-9]{8}$/;
            if(phoneNumber.match(regPhone)){
                    console.log("Correct Mobile number.");
            }else{
                    console.log("Not a valid number.");
            }

</script>
```

Output:



```
                                                         regular_expression.html:18
 ▼(2) [Array(1), Array(1)]
   ▶0: ['Namaste', index: 0, input: "Namaste sir. You didn't response my namaste.",
   ▶1: ['namaste', index: 36, input: "Namaste sir. You didn't response my namaste.",
    length: 2
   ▶[[Prototype]]: Array(0)
 ▼(2) ['Namaste', 'namaste']                             regular_expression.html:21
    0: "Namaste"
    1: "namaste"
    length: 2
   ▶[[Prototype]]: Array(0)
 Correct Mobile number.                                  regular_expression.html:27
>
```

Example 4:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Email Validation</title>
</head>
<body>
    <form>
            Email: <input type="email" name="email" id="email" /></br></br>
            <input type="button" name="btn" value="Email Validation" 
onclick="emailValidation()" />
    </form>
    <span id="v-result"></span>

    <script type="text/javascript">
            function emailValidation(){
                    var res = document.getElementById('v-result');
                    let emailPattern = /^[^-9][a-z0-9._]{3,}@[a-z]{2,}.{1}[a-z]{2,}(.[a-z]+)?$/;
                    var inputEmail = document.getElementById('email').value;
                    if (inputEmail.match(emailPattern)) {
                            res.innerHTML = "Provided email is valid.";
                    }else{
                            res.innerHTML= "Incorrect. Enter valid email.";
                    }

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

Error Handling

There are three types of errors in programming: (a) Syntax Errors, (b) Runtime Errors, and (c) Logical Errors.

- **Syntax error** – also called **parsing errors,** occur at compile time in traditional programming languages and at interpret time in JavaScript. When syntax error occurs , only the code contained within the same thread as the syntax error is affected and the rest of the code in other threads gets executed assuming nothing in them depends on the code containing the error.
- Runtime error – also called **exceptions,** occur during execution (after compilation/interpretation). Exceptions also affect the thread in which they occur, allowing other JavaScript threads to continue normal execution.
- Logical error – Logic errors can be the most difficult type of errors to track down. These errors are not the result of a syntax or runtime error. Instead, they occur when you make a mistake in the logic that drives your script and you do not get the result you expected.

The try...catch...finally Statement

The latest versions of JavaScript added exception handling capabilities. JavaScript implements the **try...catch...finally** construct as well as the **throw** operator to handle exceptions. You can **catch** programmer-generated and **runtime** exceptions, but you cannot **catch** JavaScript syntax errors.

Example: You can use **finally** block which will always execute unconditionally after the try/catch.

```
<html>
  <head>
    <script type = "text/javascript">
        function myFunc() {
          var a = 100;
          try {
            alert("Value of variable a is : " + a );
          }
          catch ( e ) {
            alert("Error: " + e.description );
          }
          finally {
            alert("Finally block will always execute!" );
          }
        }
    </script>
  </head>
  <body>
    <p>Click the following to see the result:</p>
    <form>
      <input type = "button" value = "Click Me" onclick = "myFunc();" />
    </form>
  </body>
</html>
```

The throw statement

You can use **throw** statement to raise your built-in exceptions or your customized exceptions.

Example:(validate input number)

```
<html>
<body>
  <p>Please input a number between 5 and 10:</p>
  <input id="demo" type="text">
  <button type="button" onclick="myFunction()">Test Input</button>
  <p id="p01"></p>
<script>
  function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if(x == "")  throw "empty";
    //isNaN , is Not a Number
    if(isNaN(x)) throw "not a number";
    x = Number(x);
    if(x < 5)  throw "too low";
    if(x > 10)   throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err;
  }
 }
</script>
</body>
</html>
```

The onerror() method

The **onerror** event handler was the first feature to facilitate error handling in JavaScript. The **error** event is fired on the window object whenever an exception occurs on the page.

The **onerror** event handler provides three pieces of information to identify the exact nature of the error :

- **Error message** − The same message that the browser would display for the given error
- **URL** − The file in which the error occurred
- **Line number** − The line number in the given URL that caused the error

Example:

```
<html>
  <head>
    <script type = "text/javascript">
        window.onerror = function (msg, url, line) {
          alert("Message : " + msg );
          alert("url : " + url );
          alert("Line number : " + line );
        }
    </script>
  </head>
  <body>
    <p>Click the following to see the result:</p>
    <form>
      <input type = "button" value = "Click Me" onclick = "myFunc();" />
    </form>
  </body>
</html>
```

Type conversion in JavaScript

JavaScript variables can be converted to a new variable and another data type:

- By the use of a JavaScript function
- **Automatically** by JavaScript itself

Converting Numbers to Strings

The global method ***String()*** can convert numbers to strings. Similarly ***boolean*** , and ***date*** can be converted to string.

Example:

```
<!DOCTYPE html>
<html>
<body>
     <h2>The JavaScript String() Method</h2>
     <p>The String() method can convert a number to a string.</p>
     <p id="demo"></p>
<script>
     var x = 123;
     document.getElementById("demo").innerHTML =
     String(x) + "<br>" +
     String(123) + "<br>" +
     String(100 + 23);
</script>
</body>
</html>
```

Converting Strings to Numbers

Example:

```
<script>
     var x = “123”;
     document.getElementById("demo").innerHTML =Number(x) ;
     var y=”5”; //y is a string
     var z=+y; //z is a number
     document.getElementById(‘demo1’).innerHTML=z;
     var a=”4”; //a is string
     var b=6;  //b is number
     var c=a+b;  //+ sign concatenate “4” and 6 =46  not =10
     document.getElementById(‘demo2’).innerHTML=c;
</script>
```

Chapter 5

Programming in PHP and MySQL

Some Old Questions:

a) Explain the origin, major features and arrays of PHP.

b) How is array implemented in PHP ? Differentiate between *foreach* loop and *each()* function with example.

c) Write a program in PHP to list out all prime numbers within a given range.

d) What is exception handling ? Write a program that validate email "***someone@example.com***" using exception handling with PHP script.

e) Why Regular Expression is used ? How does PHP support pattern matching ? Explain with example.

f) Write about PHP. What are the main uses of PHP in web programming? Explain different kinds of array in PHP.

g) Explain the use of PHP $_GET and $_POST variables.

h) Describe Starting , Storing and Destroying a Session. Explain with sample codes.

i) Explain about Form handling and File handling in PHP with examples.

j) Discuss the use of cookies and sessions with example scenario of each in PHP.

k) Write CRUD queries required to connect, fetch , delete and update data in PHP from MYSQL.

l) How do you connect PHP with MySQL ? Write a PHP Code that retrieves the name and password from database.

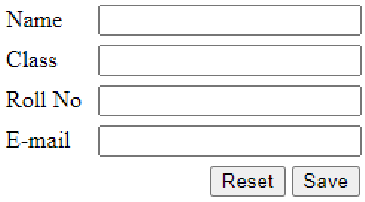
m) What are the uses of session ? Is it possible to work with sessions in PHP? Explain with an example.

n) What is type conversion? How do PHP deal with type conversion? Explain.

o) Write a server side script for user registration having input fields name, password, age, email, phone with a proper client side validation.

p) Write a PHP code that connects with the database and insert (firstname, lastname, age) in a table using, insertion query.

q) Write necessary PHP scripts for following to make connection with the database and insert form data to database table , clicked on "**Save**" button. Where "Reset" reset HTML form.



r) Write MySQL query to:
   a. Create a database named ***management*** and use it.
   b. Create a table named ***users*** with id(integer), name(string), email(string), password(string), subscriber(Boolean).
   c. Insert data in the table.
   d. Extract data from the table.

s) Short Notes On:

ii) Session and Cookies

iii) MySQL joins

Origins and Uses of PHP

- Php was developed by Rasmus Lerdorf, a member of the Apache Group, in 1994.
- PHP is an open-source product.
- PHP is an acronym for Personal Home Page, or PHP: Hypertext Preprocessor.
- PHP is used for form handling, file processing, and database access.

Overview of PHP

- PHP is a server-side scripting language whose scripts are embedded in HTML documents - Similar to JavaScript, but on the server side .
- PHP is an alternative to CGI, Active Server Pages (ASP), and Java Server Pages (JSP) .
- The PHP processor has two modes: copy (HTML) and interpret (PHP).
- It takes a PHP document file as input and produces an HTML document file.
- When the php processor finds markup code in the input file, it simply copies it to the output file.
- When the processor encounters php script in the input file, it interprets it and sends any output of the script to the input file.
- This implies that the output from a PHP script must be HTML , either of which could include embedded client-side script.
- PHP syntax is similar to that of JavaScript.
- PHP is dynamically typed, as does JavaScript. Variables are not type declared, and they have no intrinsic type.

General Syntactic Characteristics

- PHP is a server-side scripting language i.e. embedded in HTML.
- PHP scripts also are in files that are referenced by such documents.
- PHP code is embedded in document by enclosing it between <*?php* and *?*> tags.
- The include keyword is used to reference the file where php script is stored, which takes the file name as its string parameter. eg: include(“database.php”).
- Thus included file can contain markup or client-side script, as well as PHP code, but any PHP script it include must be the content of <*?php* tag .
- Every variable name begins with a dollar sign ($).
- PHP variable names are case sensitive .
- But, neither reserved words nor function names are case sensitive (no difference between while, WHILE, While, and wHile).
- Single line comments to be specified either with # or with // .
- Multiple line comments are specified with /* and */ .
- PHP statements are terminated with semicolons.
- Braces are used to form compound statements for control statements.
- Some reserved words of PHP are listed below:

And	Else	Global	Require	Virtual	Break
Elseif	If	Return	Xor	Case	Extends
Include	Static	While	Class	False	List
Switch	Continue	For	New	This	Default

Foreach	Not	True	Do	Function	Or
var					

Variables:

- There are no type declarations .
- An unassigned (unbound) variable has the value, *NULL*.
- The ***unset*** function sets a variable to NULL.
- The ***IsSet*** function is used to determine whether a variable is *NULL*.
- A variable starts with the $ sign, followed by the name of the variable.
- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Rules for PHP variables:
  - A variable starts with the $ sign, followed by the name of the variable
  - A variable name must start with a letter or the underscore character
  - A variable name cannot start with a number
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  - Variable names are case-sensitive ($age and $AGE are two different variables)

Output

- Output from a PHP script is HTML that is sent to the browser .
- HTML is sent to the browser through standard output.
- There are three ways to produce output: ***echo***, ***print***, and ***printf***.
- ***echo*** and ***print(****slower than echo****)*** take a string, but will coerce other values to strings.
  echo "whatever"; # Only one parameter
  echo("first ", $sum) # More than one
  print "Welcome to my site!"; # Only one
  print "My favorite color is"." ".$color; # More than one
- The ***printf()*** function outputs a formatted string. The arg1, arg2, ++ parameters will be inserted at percent (%) signs in the main string. This function works "step-by-step". At the first % sign, arg1 is inserted, at the second % sign, arg2 is inserted, etc.
  Example:

```
<!DOCTYPE html>
<html>
<body>
        <?php
                $txt = "W3Schools.com";
                echo "I love $txt!";
                $x=5;
                $y=4:
                $z=$x+$Y;
                echo "The sum is $Z"."<br>";
                echo "The sum is  "."$z";
                $str = "nine";
                printf("The sum of %u and %u is %s.",$x,$y,$str);
                print "Hello world!<br>";
        ?>
```

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

- ***fprintf()*** – used to write formatted string to output stream (file and database).

  Example:

  ```
  <?php
          $str = "Frontend Developer.";
          $salary = 40000;
          $file = fopen("abc.txt","w");
          echo fprintf($file,"%s monthly salary is %u.",$str,$salary);
  ?>
  ```

  Note: Above results output 43 and string written to the file.

PHP Data Types

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
  - String – A string is a sequence of characters, like "Hello world!". A string can be any text inside quotes. You can use single or double quotes.
  - Integer – An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
  - Float (floating point numbers - also called double) – a float (floating point number) is a number with a decimal point or a number in exponential form.
  - Boolean – a Boolean represents two possible states: TRUE or FALSE.
  - Array – an array stores multiple values in one single variable.
  - Object – An object is a data type which stores data and information on how to process that data. An object must be explicitly declared.
  - NULL – Null is a special data type which can have only one value: *NULL*.
  - Resource – The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP. A common example of using the resource data type is a database call.

    Example:

    ```
    <!DOCTYPE html>
    <html>
    <body>
       <?php
               //String
               $x = "Hello world!";
               $y = 'Hello world!';
               echo $x;
               echo "<br>";
               echo $y;
               //Integer
               $x = 5985;
               var_dump($x);
               //The PHP var_dump() function returns the data type and value
               //Floating point
               $y = 10.365;
               var_dump($y);
               //Array
               $cars = array("Volvo","BMW","Toyota");
               var_dump($cars);
               //class and object
    ```

```
                class Car {
                         function Car() {
                        $this->model = "VW";
                        }
                        }
                        // create an object
                        $typ = new Car();
                        // show object properties
                        echo $typ->model;
    ?>
    </body>
    </html>
```

String Method

- ***strlen()*** – returns the length of string.
- ***str_word_count()*** – function counts the number of words in a string.
- ***strrev()*** – reverse the string.
- ***strpos()*** – function searches for a specific text within a string.
- ***str_replace()*** – function replaces some characters with some other characters in a string.
- ***strcmp()*** – The strcmp() function compares two strings.
- ***str_replace(search,replace,subject)*** – replace existing value by new one**.**
- ***strtoupper()*** – transform to uppercase.
- ***Strtolower()*** – transform to lowercase.
- ***Lcfirst()*** – transform first character to lowercase.
- ***Ucfirst()*** – transform first character to uppercase.
- ***Ucwords()*** – transform first character of each word to uppercase.

  Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
        echo strlen("Hello world!");
        echo str_word_count("Hello world!");
        echo strrev("Hello world!");
        echo strpos("Hello world!", "world");
        echo str_replace("world", "Dolly", "Hello world!");
        echo strcmp("Hello world!","Hello world!");
        echo strtoupper("to uppercase.");
        echo strtolower("TO LOWERCASE");
        echo lcfirst("HELLO");
        echo ucfirst("hello");
        echo ucwords("namaste nepal.");
?>
        <p>If strcmp() function returns 0, the two strings are equal.</p>
</body>
</html>
```

PHP Constants

- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- To create a constant, use the ***define()*** function.

  Syntax:

  define(name, value, case-insensitive);

  Parameters:

  - *name*: Specifies the name of the constant
  - *value*: Specifies the value of the constant
  - *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

  Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
// case-sensitive constant name
        define("GREETING", "Welcome to PHP class!");
        echo GREETING;
// case-insensitive constant name
        define("WPT", "This is web technology class.", true);
        echo wpt;
?>
</body>
</html>
```

PHP Date/Time

Date/Time function allowed you to get date and time from server where your script runs. You can set date and time as needed in several ways. There are a lots of function related date and time that you can implement them in your project. Among which here we have discuss on date() function to format local date and time and return formatted date string.

Syntax:

date(format, timestamp);

Format and description:

Y – A four digit representation of a year.

y – A two digit representation of a year.

m – a numeric representation of a month(01 to 12).

n – a numeric representation of a month, without leading zeros(1 to 12).

M – a short textual representation of a month(Jan to Dec).

F – a full textual representation of a month(January to December).

W – the iso-8601 week number of year (weeks starting on Monday).

w – a numeric representation of the day (0 for Sunday, 6 for Saturday).

t – the number of days in the given month.

z – the day of the year (from 0 through 365).

d – the day of the month (from 01 to 31).

D – a textual representation of a day(three letters).

j – the day of the month without leading zeros(1 to 31).

l – a full textual representation of a week day.

h – 12-hour format of an hour(01 to 12)

H – 24-hour format of an hour (00 to 23).

i – minutes with leading zeros(00 to 59).

s – seconds, with leading zeros(00 to 59).

a – lowercase am or pm.

A – uppercase AM or PM.

timestamp *is optional.*

Example:

```
<?php
	/*
		Lets get today's date and time.
		First we have to set our time zone i.e. asia/kathmandu.
	*/
		//check server time zone
		 $checkTimeZone = date_default_timezone_get();
		 echo $checkTimeZone;

		 //Now set to our time zone
		 date_default_timezone_set('asia/kathmandu');
		 //Now check time zone
		 $localTimeZone = date_default_timezone_get();
		 echo "<br>";
		 echo $localTimeZone;

		 //finally lets get today's date and time
		 $dateAndTime = date('Y/m/d::h:i:s A l');
		 echo "<br>";
		 echo $dateAndTime;
?>
```

Output: 2022/01/26::08:13:18 PM Wednesday

Control/Conditional statement

- Conditional statements are used to perform different actions based on different conditions.
- In PHP we have the following conditional statements:
    - ***if*** statement - executes some code if one condition is true
      Syntax:
      ```
      if(condition){
        //code to be executed if condition is true
      }
      ```
    - ***if…else*** statement - executes some code if a condition is true and another code if that condition is false.
      Syntax:
      ```
      if(condition){
        //code to be executed if condition is true
      }
      else{
        //code to be executed if condition is false
      }
      ```
    - ***if…elseif…else*** statement - executes different codes for more than two conditions.
      Syntax:
      ```
      if(condition){
        //code to be executed if this condition is true
      }
      elseif(condition){
        //code to be executed if this condition is true
      }
      else{
        //code to be executed if all condition is false
      }
      ```
    - ***switch*** statement - selects one of many blocks of code to be executed.
      Syntax:
      ```
      switch(n){
        case label1:
                //code to be executed if label1=n;
                break;
        case label2:
                //code to be executed if label2=n;
                break;
      ………....
      default:
        //code to executed if n is different from all labels;
      }
      ```

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
```

```
            $t = date("H");
            echo "<p>The hour (of the server) is " . $t;
            echo ", and will give the following message:</p>";
            if ($t < "10") {
             echo "Have a good morning!";
            } elseif ($t < "20") {
            echo "Have a good day!";
            } else {
            echo "Have a good night!";
            }
?>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
            $favcolor = "red";
            switch ($favcolor) {
            case "red":
                        echo "Your favorite color is red!";
                         break;
            case "blue":
                        echo "Your favorite color is blue!";
                        break;
            case "green":
                        echo "Your favorite color is green!";
                         break;
            default:
                        echo "Your favorite color is neither red, blue, nor green!";
}
?>
</body>
</html>
```

Loop Statements

- Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

  The for loop statement

  The for statement is used when you know how many times you want to execute a statement or a block of statements.

  Flowchart:

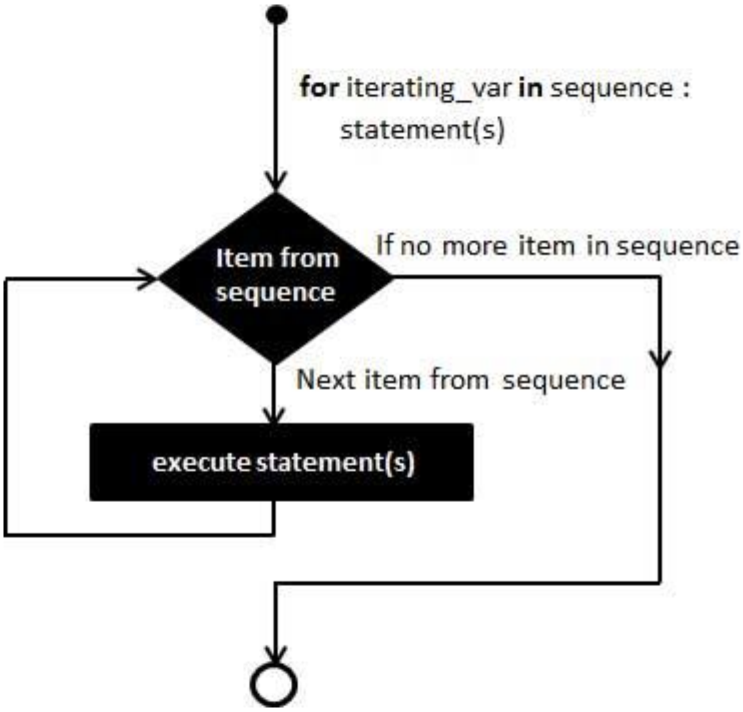


figure – flowchart of for loop

Syntax:

```
for(initialization; test-condition; iteration-statement){
    //statement(s) to be executed if expression is true
}
```

Example:

```
<html>
  <body>
    <?php
         $a = 0;
         $b = 0;
          for( $i = 0; $i<5; $i++ ) {
         $a += 10;
         $b += 5;
         }
         echo ("At the end of the loop a = $a and b = $b" );
    ?>
  </body>
</html>
```

The While loop statement

Syntax:

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

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
```

```
            $x = 1;
            while($x <= 5) {
                        echo "The number is: $x <br>";
                        $x++;
            }
 ?>
 </body>
 </html>
```

The PHP do...while Loop

Syntax:

do{

// code to be executed

} while(condition is true)

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
            $x = 1;
            do {
                         echo "The number is: $x <br>";
                        $x++;
            } while ($x <= 5);
?>
</body>
</html>
```

The PHP foreach Loop

The ***foreach*** loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax:

foreach($array as $value){

//code to be executed

}

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
            $colors = array("red", "green", "blue", "yellow");
            foreach ($colors as $value) {
            echo "$value <br>";
            }
?>
</body>
</html>
```

PHP Function(General/anonymous/arrow)

The real power of PHP comes from its functions; it has more than 1000 built-in functions. But here you study about user defined function. We can create our own functions .

- A function is a block of statements and is reusable in a program.

- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.
- A user-defined function declaration starts with the word ***function***:

Syntax:

```
    function function_Name(){
            //code to be executed
    }
```

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
    function writeMsg() {
    echo "Hello world!";
    }
    writeMsg();
    //function having arguments/parameters
    function familyName($fname, $year) {
    echo "$fname was Born in $year <br>";
    }
    familyName("Ram", "1975");
    familyName("Shyam", "1978");
    //Default argument value
    function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
    }
    setHeight(350);
    setHeight(); // will use the default value of 50
    //Function – returning values
    function sum($x, $y) {
    $z = $x + $y;
    return $z;
    }
    echo "The sum of 5 + 10 = " . sum(5, 10) . "<br>";
    echo "The sum of 7 + 13 = " . sum(7, 13) . "<br>";
?>
</body>
</html>
```

anonymous function: A function without a name is called an anonymous function. If assign to any variable then we need to call/invoke the function otherwise not.

Syntax: Depend how your are using this function.

```
(function(){
        //........
})
```

Example:

```
<?php
    //no function name
    $af = function(){
            echo "Anonymous Function";
    };
    $af(); //In this case well call/invoke anonymous function
    echo "<br>";
    //anonymous function variable assignment
    $welcome = function($name){
                printf("Hello,Welcome %s.",$name);
                echo "<br>";
    };
    $welcome("Ram");
    $welcome("PHP developer");

    //anonymous function as arguments
    $points = array(3,4,9,8);
    print_r(array_map(function($a){
                        return $a*$a;
                },$arr));
?>
```

Arrow Function: Introduced in PHP 7.4 version . More concise syntax for anonymous functions. Both anonymous function and arrow functions are implemented using the Closure class. The ***fn*** keyword is used to create arrow functions. Arrow functions have access to all variables from the scope in which they were created.

Syntax:

fn(arguments) => expression to be returned;

Example:

```
<?php
        //arrow function
        $fnum = 10;
        $add = fn($snum) => $fnum*$snum;
        echo $add(5)."<br>";
        var_dump($add(3)); //var_dump() function
        echo "<br>";
        var_export($add(4));//var_export() function

        //arrow function with in function as callback function
        $points = array(3,4,9,8);
        echo "<br>";
        print_r(array_map(fn($a) => $a*$a,$arr)); //used as callback function
        echo "<br>";
        var_dump(array_map(fn($a) => $a*$a,$arr));             //var_dump() function
        echo "<br>";
        var_export(array_map(fn($a) => $a*$a,$arr));//var_export() function

?>
```

var_dump() – dump information about a variable i.e. holds type and value of a variable.

var_export() – returns or outputs structured information about a variable.

Array in PHP

- An array is a special variable, which can hold more than one value at a time.
- An array can hold many values under a single name, and you can access the values by referring to an index number.
- In PHP, the ***array()*** function is used to create an array.
- In PHP, there are three types of arrays:
    - **Indexed arrays** - Arrays with a numeric index
    - **Associative arrays** - Arrays with named keys
    - **Multidimensional arrays -** Arrays containing one or more arrays

PHP Indexed Arrays

- There are two ways to create indexed arrays:
- The index can be assigned automatically (index always starts at 0), like this.
  Syntax:
  $variableName = array(elemen

Example:

```
<?php
   $cars = array("Volvo", "BMW", "Toyota");
   echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
   echo count($cars);  //to count length of an array

   //using for loop for indexed array
   $vehicle= array("car", "bike", "bus");
   $arrlength = count($vehicle);
   for($x = 0; $x < $arrlength; $x++) {
      echo $vehicle[$x];
      echo "<br>";
   }
?>
```

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

```
            $age=array("Ram"=>34, "Shyam"=>54,"Hari"=>50);  OR
            $age['Peter'] = "35";
            $age['Ben'] = "37";
            $age['Joe'] = "43";
```

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
            $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
            echo "Peter is " . $age['Peter'] . " years old.";
            //using foreach loop
            foreach($age as $x => $x_value) {
            echo "Key=" . $x . ", Value=" . $x_value;
            echo "<br>";
}
?>
</body>
</html>
```

Multidimensional Arrays

An array containing one or more arrays and values are accessed using multiple indices. A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example:

```
<?php
            $cars = array
            (
             array("Volvo",22,18),
            array("BMW",15,13),
            array("Saab",5,2),
            array("Land Rover",17,15)
            );
            for ($row = 0; $row < 4; $row++) {
            echo "<p><b>Row number $row</b></p>";
            echo "<ul>";
            for ($col = 0; $col < 3; $col++) {
            echo "<li>".$cars[$row][$col]."</li>";
            }
             echo "</ul>";
            }
?>
```

Array sorting methods

The elements in an array can be sorted in alphabetical or numerical order, descending or ascending. The sort function , which takes an array as a parameter , sorts the values in the array, replacing the keys with the numeric keys 0,1,2,. .

PHP - Sort Functions For Arrays:

- ***sort()*** – sort arrays in ascending order
- ***rsort()*** – sort arrays in descending order
- ***asort()*** – sort associative arrays in ascending order, according to the value
- ***ksort()*** – sort associative arrays in ascending order, according to the key
- ***arsort()*** – sort associative arrays in descending order, according to the value
- ***krsort()*** – sort associative arrays in descending order, according to the key

Example: **Sort Array in Ascending Order - sort():**

```
<?php
        $cars = array("Volvo", "BMW", "Toyota");
        sort($cars);
        $clength = count($cars);
        for($x = 0; $x < $clength; $x++) {
        echo $cars[$x];
        echo "<br>";
        }
?>
</body>
</html>
```

Example: **Sort Array (Ascending Order), According to Value - asort():**

```
 <!DOCTYPE html>
 <html>
 <body>
 <?php
        $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
        asort($age);
        foreach($age as $x => $x_value) {
                 echo "Key=" . $x . ", Value=" . $x_value;
                echo "<br>";
        }
 ?>
 </body>
 </html>
```

Example: **Sort Array (Ascending Order), According to Value - ksort():**

```
<!DOCTYPE html>
<html>
<body>
<?php
        $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
        ksort($age);
        foreach($age as $x => $x_value) {
                echo "Key=" . $x . ", Value=" . $x_value;
                echo "<br>";
        }
?>
</body>
</html>
```

Example: **Sort Array (Descending Order), According to Key - krsort():**

```
 <!DOCTYPE html>
 <html>
 <body>
 <?php
        $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
        krsort($age);
        foreach($age as $x => $x_value) {
                echo "Key=" . $x . ", Value=" . $x_value;
                echo "<br>";
        }
 ?>
 </body>
 </html>
```

Some more array methods:

print_r()

The print_r() function is a built-in function in PHP and is used to print or display information stored in a variable.

Syntax:

print_r(variable, return)

Parameters: This function accepts two parameters as shown in above syntax and described below.

1. **variable**: This parameter specifies the variable to be printed and is a mandatory parameter.
2. **return**: This an optional parameter. This parameter is of boolean type whose default value is FALSE and is printable. If this parameter is set to TRUE then the **print_r()** function will not print any thing.

   Example:

```
<?php
  $color = array("red", "green", "blue");
  print_r($color);

  echo "<br>";
```

```
        $person = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
        print_r($person,true); //no output will be printed
    ?>
```

each()

- is used to returns a two -element array consisting of the key and value of the current element.

  Syntax:

  each(array)

  Example:

```
    <?php
            $salaries=array("Shyam"=>4444,"Ram"=>55555,"Hari"=>6666);
            while($employee=each($salaries)){
                    $name=$employee["key"];
                    $salary=$employee["value"];
                    print("The salary of $name is $salary. <br/>");
    }
    ?>
```

array_slice()

Return selected part of an array i.e. make new array from array.

Syntax:

array_slice(array,start,length,preserve)

array – specifies defined array .Required

start – slice starting numeric value i.e. index of array(0,1,2,…). If this value is –ve , function will start slicing from the last element. -3 ,start from 3rd last element of the array.

length – Optional, numeric value .Specifies the length sliced array. If this not set , all elements from start will be sliced.

preserve – Optional. true – preserve keys and false – reset keys, default value.

Example:

```
    <?php
                echo "<br><b>Indexed Array slicing:</b>";
            // Preserve parameter set to false(default):
                $color=array("red","green","blue","yellow","brown");
                print_r($color);
                echo "<br><b>Sliced started from:[2],of length:2, false(default):</b> <br>";
                print_r(array_slice($color,2,2));
            // Preserve parameter set to true:
                echo "<br><b>Sliced started from:[2],of length:2,true:</b> <br>";
                print_r(array_slice($color,2,2,true));
                echo"<br>";
            // without length start at -ve value
                echo "<br><b>Sliced started from last 3 element i.e. -3:</b> <br>";
                $color=array("red","green","blue","yellow","brown");
                print_r(array_slice($color,-3));
            //associative array slicing
                echo "<br><b>Associative array slicing:</b>";
                $aColor = array('r'=> "red",'b' => "blue",'g' => "green",'y' => "yellow");
                print_r($aColor);
                echo "<br><b>Sliced all elements started from:[2]:</b> <br>";
                print_r(array_slice($aColor,2));
                echo"<br>";
```

```
?>
```

Output:

```
Indexed Array slicing:Array ( [0] => red [1] => green [2] => blue [3] => yellow [4] => brown )
Sliced started from:[2],of length:2, false(default):
Array ( [0] => blue [1] => yellow )
Sliced started from:[2],of length:2,true:
Array ( [2] => blue [3] => yellow )

Sliced started from last 3 element i.e. -3:
Array ( [0] => blue [1] => yellow [2] => brown )
Associative array slicing:Array ( [r] => red [b] => blue [g] => green [y] => yellow )
Sliced all element started from:[2]:
Array ( [g] => green [y] => yellow )
```

array_udiff() :

The *array_udiff()* function compares the values of two or more arrays, and returns the differences.

Note: This function uses a user-defined function to compare the values!

This function compares the values of two (or more) arrays, and return an array that contains the entries from *array1* that are not present in *array2* or *array3*, etc.

Example:

```
<?php
function myfunction($a,$b)
{
if ($b===$a)
  {
  return 0;
  }
  return ($b>$a)?-1:1;
}
$a1=array("a"=>"red","b"=>"green","c"=>"blue");
$a2=array("a"=>"blue","b"=>"black","e"=>"blue");
$result=array_udiff($a2,$a1,"myfunction");
print_r($result);
?>
```

Output:

```
Array ( [b] => black )
```

array_splice():

The array_splice() function removes selected elements from an array and replaces it with new elements. The function also returns an array with the removed elements. If set length=0 , not any element be removed and insert elements from start.

Syntax:

array_splice(array_name,start,length,array_name_to_insert);

Example:

```
<?php
        $colors=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
        $additionalColor=array("a"=>"purple","b"=>"orange");

                echo "<br>";
                echo "color array:";
                print_r($colors);
                echo "<br>";
                echo "additionalColor array:";
                print_r($additionalColor);

        //remove elements from array
                echo "<br>";
                echo "Splice array from start:0 ,total elements: 2 ::";
                print_r(array_splice($colors,0,2));
                echo "<br>";
                echo "colors array after spliced:";
                print_r($colors);

        //add elements to colors array
                echo "<br>";
                //Splice array from start=1,remove 0 elements, and add elements of additionalColor array.
                array_splice($colors,1,0,$additionalColor);
                echo "<br>";
                echo "After Splice:";
                print_r($colors);
?>
```

Some Array Methods In PHP , Example:

```
<?php
        /*
                Some Array methods
        */
                //print_r()
                $arr = array(1,2,3,6,9); //indexed based array
                $arr1 = array('b' =>"Bikash",'c' =>'Chandra','d' => "Deepak"); //associative array(key =>
value in pair)
                print_r($arr);
                echo "<br>";
                print_r($arr1);
                echo "<br>";

                //array_map()
                print_r(array_map("elementProduct",$arr));
                function elementProduct($a){
                        return $a*$a;
                }
                //array_merge()
                $arr2 = array(4,40,50);
                $color = array('l' => "Blue",'r'=>"red",'g' => "green");
                echo "<br>";
                print_r(array_merge($arr2,$arr));
                echo "<br>";
                print_r(array_merge($color,$arr1));

                //array_pop()
                echo "<br>";
                print_r(array_pop($arr2));
                print_r($arr2);

                //array_push()
                echo "<br>";
                print_r(array_push($arr,80));
                echo "<br>";
                print_r($arr);

                //array_shift()
                echo "<br>";
                print_r(array_shift($arr));
                echo "<br>";
                print_r($arr);

                //array_unshift()
                echo "<br>";
                print_r(array_unshift($arr,100));
                echo "<br>";
                print_r($arr);

                //array_sum()
                echo "<br>";
                print_r(array_sum($arr));

                //array_product()
                echo "<br>";
                print_r(array_product($arr));

                //array_search()
                echo "<br>";
                print_r(array_search(200,$arr));
                echo "<br>";
                print_r(array_search('red',$color));

?>
```

Form Handling

Forms could be handled by the same document that creates the form, but that may be confusing. It does not matter whether GET or POST method is used to transmit the form data. PHP builds a variable for each form element with the same name as the element.

Example 1: Form containing html source code ,file name *form.php*

```
<!DOCTYPE HTML>
<html>
<body>
        <form action="welcome.php" method="post">
        Name: <input type="text" name="name"><br>
        E-mail: <input type="text" name="email"><br>
        <input type="submit">
        </form>
</body>
</html>
```

The php file that response to above through action attribute is *welcome.php*. I.e. given below:

```
<html>
<body>
		Welcome <?php echo $_POST["name"]; ?><br>
		Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

Example 2: Sending Data from form to database dynamically.

```
<!-- formHandling.php -->
<!DOCTYPE html>
<html>
<head>
		<meta charset="utf-8">
		<meta name="viewport" content="width=device-width, initial-scale=1">
		<title>Form Handling</title>
		<script src='./effect.js'>

		</script>
		<style type="text/css">
				form{
						/*background-color: #d33030;*/
						padding: 50px 100px;
						width: 50%;
						border-radius: 30px;
						border: 1px solid black;
				}
				#btn{
						background:  none;
						border:  1px solid #393939;
						padding: 6px 12px;
						border-radius: 6px;
				}
		</style>

</head>
<body>
		<form action="response.php" method="post" target="_blank" >
				Full Name: <input type="text" name="cname" value="Ram" required /> </br></br>
				Email:<input type="email" name="cemail" value="ram@gmail.com" required /></br></br>
				Address: <input type="text" name="caddress" value="Kathmandu" required /></br></br>
				<input type="submit" name="imply" value="Send Data" id="btn"
onclick="changeButtonColor(this)" />
		</form>
</body>
</html>

<!-- response.php -->
<?php
```

```
    if(isset($_POST['imply'])){
        $host = 'localhost';
        $user = 'root';
        $pass = '';
        $db = 'timro_hamro_fashion_ghar';

        $name = $_POST['cname'];
        $email = $_POST['cemail'];
        $address = $_POST['caddress'];



        $conn = mysqli_connect($host,$user,$pass,$db);
        if(!$conn){
            die("Database not found !!!");
        }

        $sql = "INSERT INTO tbl_cus(name,email,address)
VALUES('".$name."','".$email."','".$address."')";
        $query = mysqli_query($conn,$sql);
        if(!$query){
            die("Unable to Insert Record!!!");
        }else{
            echo "Record Inserted Successfully.";
        }
        mysqli_close($conn);
    }
?>
// effect.js
function changeButtonColor(obj){
    obj.style.backgroundColor = "blue";
    obj.style.border = "none";
    obj.style.padding = "10px";
    obj.style.color = "#FFFFFF";
    console.log("hello");

}
```

File Handling:

File handling is an important part of any web application. Here we will explain following functions related to files:

- Opening a file
- Reading a file
- Writing a file
- Closing a file

Opening and Closing Files

The PHP ***fopen()*** function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate. Files modes can be specified as one of the six options in this table.

Sr.No	Mode & Purpose
1	r Opens the file for reading only. Places the file pointer at the beginning of the file.
2	r+ Opens the file for reading and writing. Places the file pointer at the beginning of the file.
3	**w** Opens the file for writing only. Places the file pointer at the beginning of the file and truncates the file to zero length. If files does not exist then it attempts to create a file.
4	**w+** Opens the file for reading and writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
5	**a** Opens the file for writing only. Places the file pointer at the end of the file.If files does not exist then it attempts to create a file.
6	**a+** Opens the file for reading and writing only.Places the file pointer at the end of the file.If files does not exist then it attempts to create a file.

Reading a file

Once a file is opened using **fopen()** function it can be read with a function called ***fread()***. This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.

The files length can be found using the ***filesize()*** function which takes the file name as its argument and returns the size of the file expressed in bytes.

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
        $myfile = fopen("student.txt", "r") or die("Unable to open file!");
        echo fread($myfile,filesize("webdictionary.txt"));
        fclose($myfile);
?>
</body>
</html>
```

Example: Reading single line – *fgets()* function

```
//The fgetc() function is used to read single line from a file , named student.txt
<!DOCTYPE html>
<html>
<body>
<?php
        $myfile = fopen("student.txt", "r") or die("Unable to open file!");
        echo fgets($myfile);
        fclose($myfile);
?>

</body>
</html>
```

Example: Reading single character – *fgetc()* function

```
//The fgetc() function is used to read single character from a file , named student.txt
<!DOCTYPE html>
<html>
<body>
<?php
        $myfile = fopen("student.txt", "r") or die("Unable to open file!");
        // Output one character until end-of-file
        while(!feof($myfile)) {
                echo fgetc($myfile);
        }
        fclose($myfile);
?>
</body>
</html>
```

PHP Check End-Of-File – feof()

The ***feof()*** function checks if the "end-of-file" (EOF) has been reached. The ***feof()*** function is useful for looping through data of unknown length.

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
        $myfile = fopen("student.txt", "r") or die("Unable to open file!");
        // Output one line until end-of-file
        while(!feof($myfile)) {
                echo fgets($myfile) . "<br>";
        }
        fclose($myfile);
?>
</body>
</html>
```

Some methods of file handling:

***Note**: To see effect of functions, un-comment to individual file handling function.*

```
<?php
        /*
                FILE HANDLING FUNCTIONS:
                readfile(),file_exists(),filesize(),fopen(),fread(),fclose()
        */
                // if(file_exists("newFile.txt")){
                //        echo readfile("newFile.txt");
                //        echo "<br>";
                //        echo filesize("newFile.txt");
                // }else{
                //        echo "File does not exists!!!";
                // }

        //fopen(filename,mode) with read mode
                // $newFile = fopen("newFile.txt", "r") or die("Unable to open.");
                // $fileRead = fread($newFile, filesize("newFile.txt"));
                // echo $fileRead;
                // fclose($newFile);
        //mode r+(reading and writing)
                // $nFile = fopen("newFile.txt", "r+");
                // fwrite($nFile, "This is not so cool.");

                // echo fread($nFile, filesize("newFile.txt"));
                //fclose($nFile);
        //mode w
                // $newWriteFile = fopen("welcome.txt", "w");
                // fwrite($newWriteFile, "Welcome to our class.");
                // fclose($newWriteFile);
                // readfile("welcome.txt");

        //mode a
                // $aFile = fopen("welcome.txt", "a");
                // fwrite($aFile, "Lets start.");
                // fclose($aFile);

                // readfile("welcome.txt");
        //copy();
                // copy("welcome.txt", "newFile.txt");
        //rename()
                // rename("newFile.txt", "copyFile");
        // mkdir("../abc");
        // rmdir("../abc");
        // unlink("copyFile.txt");
        // delete("welcome.txt");
        // echo realpath("welcome.txt");
                $path = realpath("welcome.txt");
        // print_r(pathinfo($path,PATHINFO_FILENAME));
                // echo dirname($path,4);
                echo basename($path,".txt");

?>
```

Writing a File

A new file can be written or text can be appended to an existing file using the PHP **fwrite()** function. This function requires two arguments specifying a **file pointer** and the string of data that is to be written.

Example:

```
<?php
        $myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
        $txt = "John Doe\n";
        fwrite($myfile, $txt);
        $txt = "Jane Doe\n";
        fwrite($myfile, $txt);
        fclose($myfile);
?>
```

Cookies in PHP

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies. There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser).

Setting Cookies with PHP

PHP provided ***setcookie()*** function to set a cookie. This function requires upto six arguments and should be called before <html> tag. For each cookie this function has to be called separately.

- **Name** − This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value** − This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** − This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path** − This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain** − This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security** − This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Example:

```
<!DOCTYPE html>
<?php
    $cookie_name = "user";
    $cookie_value = "Rajech Chaudhary";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
    // 86400 = 1 day
?>
<html>
<body>
<?php
    if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
    } else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
    }
?>
    <p><strong>Note:</strong> You might have to reload the page to see the value  of the
cookie.</p>
</body>
</html>
```

Delete cookies

To delete a cookie, use the ***setcookie()*** function with an expiration date in the past:

Example:

```
<!DOCTYPE html>
<?php
    // set the expiration date to one hour ago
    setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
    echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

Check if Cookies are Enabled

Example:

```
<!DOCTYPE html>
<?php
            setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>
```

```
<?php
        if(count($_COOKIE) > 0) {
                echo "Cookies are enabled.";
        } else {
                echo "Cookies are disabled.";
        }
?>
</body>
</html>
```

Example:*The isset() function is used to check if a cookie is set or not .*

```
<!DOCTYPE html>
<html>
<body>
<?php
        if(isset($_COOKIE["name"])) {
                echo "Welcome".$_COOKIE["name"]."<br>";
        } else {
                echo "Sorry ...Not recognized"."<br>";
        }
?>
</body>
</html>
```

Note: cookies can be modified by changing their name and values.

Session

- PHP Session is an alternative way to make data accessible across the various pages of an entire website.
- A session creates a file in a temporary directory on the server where registered session variables and their values are stored.
- This data will be available to all pages on the site during that visit.
- The location of the temporary file is determined by a setting in the ***php.ini*** file called ***session.save_path***.
  When a session is started following things happen :
  - PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
  - A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
  - A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.
- When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file.

- A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Starting a PHP Session

A PHP session is easily started by making a call to the ***session_start()*** function. This function first checks if a session is already started and if none is started then it starts one. Session variables are stored in associative array called ***$_SESSION[]***. These variables can be accessed during lifetime of a session. Make use of **isset()** function to check if session variable is already set or not.

Example: *The following example starts a session then register a variable called* ***counter*** *that is incremented each time the page is visited during the session.*

```
<?php
             session_start();
            if( isset( $_SESSION['counter'] ) ) {
                        $_SESSION['counter'] += 1;
            }else {
                        $_SESSION['counter'] = 1;
            }
                        $msg = "You have visited this page ".  $_SESSION['counter'];
                        $msg .= "in this session.";
?>
<html>
  <head>
    <title>Setting up a PHP session</title>
  </head>
  <body>
            <?php  echo ( $msg ); ?>
  </body>
</html>
```

Destroying a PHP Session

A PHP session can be destroyed by ***session_destroy()*** function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use ***unset()*** function to unset a session variable.

Example: ***To unset a single variable***

```
<?php
            unset($_SESSION['counter']);
?>
```

Example: ***To destroy all session variables***

```
<?php
            session_destroy();
?>
```

Error Handling

Error handling is the process of catching errors raised by your program and then taking appropriate action. If you would handle errors properly then it may lead to many unforeseen consequences.

Example: ***Using die() function***

```
<?php
        if(!file_exists("/tmp/test.txt")) {
                die("File not found");
        }else {
                $file = fopen("/tmp/test.txt","r");
                print "Opend file sucessfully";
        }
  // Test of the code here.
?>
```

Creating a Custom Error Handler

Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP. This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context).

Syntax:

error_function(error_level,error_message,error_file,error_line,error_context);

Example:

```
<?php
      //error handler function
      function customError($errno, $errstr) {
                echo "<b>Error:</b> [$errno] $errstr<br>";
                echo "Ending Script";
       die();
      }
      //set error handler
                set_error_handler("customError",E_USER_WARNING);
      //trigger error
                $test=2;
      if ($test>=1) {
      trigger_error("Value must be 1 or below",E_USER_WARNING);
}     ?>
```

Note: A *trigger_error()* function is useful to trigger errors when an illegal input occurs. Possible error types:

- ***E_USER_ERROR*** - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted.
- ***E_USER_WARNING*** - Non-fatal user-generated run-time warning. Execution of the script is not halted.
- ***E_USER_NOTICE*** - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally.

Exception Handling

Exceptions are important and provides a better control over error handling. Lets explain there new keyword related to exceptions.

- **Try** − A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown".
- **Throw** − This is how you trigger an exception. Each "throw" must have at least one "catch".
- **Catch** − A "catch" block retrieves an exception and creates an object containing the exception information.

Example:

```
<?php
        //create function with an exception
        function checkNum($number) {
                 if($number>1) {
                 throw new Exception("Value must be 1 or below");
                }
                return true;
        }
        //trigger exception in a "try" block
        try {
        checkNum(2);
        //If the exception is thrown, this text will not be shown
                echo 'If you see this, the number is 1 or below';
        }
        //catch exception
        catch(Exception $e) {
                echo 'Message: ' .$e->getMessage();
        }
?>
```

In the above example $e->getMessage function is used to get error message.

Creating a Custom Exception Class

To create a custom exception handler you must create a special class with functions that can be called when an exception occurs in PHP. The class must be an extension of the exception class.

Example:

```
<?php
class customException extends Exception {
        public function errorMessage() {
                 //error message
                $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
                .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
                return $errorMsg;
        }
}
        $email = "someone@example.com";

        try {
         //check if
                if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
        //throw exception if email is not valid
         throw new customException($email);
        }
 //check for "example" in mail address
        if(strpos($email, "example") !== FALSE) {
                throw new Exception("$email is an example e-mail");
        }
        }
        catch (customException $e) {
                echo $e->errorMessage();
        }

        catch(Exception $e) {
                echo $e->getMessage();
        }
?>
```

Example Explained:

The code above tests two conditions and throws an exception if any of the conditions are not met:

1. The ***customException()*** class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class.
2. The ***errorMessage()*** function is created. This function returns an error message if an e-mail address is invalid.
3. The ***$email*** variable is set to a string that is a valid e-mail address, but contains the string "example".
4. The ***"try"*** block is executed and an exception is not thrown on the first condition.
5. The second condition triggers an exception since the e-mail contains the string "example".
6. The ***"catch"*** block catches the exception and displays the correct error message.

MySQL Database

- MySQL is the most popular database system used with PHP.
- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

MySQL connect

Before we can access data in the MySQL database, we need to be able to connect to the server:

Example:

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    // Create connection
```

```
        $conn = new mysqli($servername, $username, $password);
        // Check connection
        if ($conn->connect_error) {
                die("Connection failed: " . $conn->connect_error);
        }
        echo "Connected successfully";
?>
```

MySQL create Database

Example: The following example creates database named db_student.

```
<?php
        $servername = "localhost";
        $username = "root";
        $password = "";
// Create connection
        $conn = mysqli_connect($servername, $username, $password);
// Check connection
        if (!$conn) {
                die("Connection failed: " . mysqli_connect_error());
        }
// Create database
        $sql = "CREATE DATABASE db_student";
        if (mysqli_query($conn, $sql)) {
                echo "Database created successfully";
        } else {
                echo "Error creating database: " . mysqli_error($conn);
        }
mysqli_close($conn);
?>
```

MySQL create Table

Example: The following example create table named tbl_student with five column : "*id*", "*firstName*", "*lastName*", "*email*" and "*address*".

```
<?php
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "db_student";
// Create connection
        $conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
        if (!$conn) {
                die("Connection failed: " . mysqli_connect_error());
        }
// sql to create table
        $sql = "CREATE TABLE tbl_student(id INT(6) UNSIGNED AUTO_INCREMENT        PRIMARY
KEY, firstName VARCHAR(30) NOT NULL, lastName VARCHAR(30) NOT        NULL, email
VARCHAR(50),address VARCHAR(30) NOT NULL)";

if (mysqli_query($conn, $sql)) {
        echo "Table tbl_student created successfully";
} else {
  echo "Error creating table: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

MySQL Insert Data

Example: The following example add a new record in table *tbl_student*.

```
<?php
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "db_student";
// Create connection
        $conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
        if (!$conn) {
                die("Connection failed: " . mysqli_connect_error());
        }
        $sql = "INSERT INTO tbl_student (firstname, lastname, email)VALUES ('John',       'Doe',
'john@example.com', 'kathmandu')";

        if (mysqli_query($conn, $sql)) {
                echo "New record created successfully";
        } else {
                echo "Error: " . $sql . "<br>" . mysqli_error($conn);
        }

        mysqli_close($conn);
?>
```

MySQL Select Data

Example: The following example selects the *id*, *firstname* and *lastname* columns from the *tbl_student* table and displays it on the page:

```
<?php
        $servername = "localhost";
        $username = "username";
        $password = "";
        $dbname = "db_student";
// Create connection
        $conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
        if (!$conn) {
                die("Connection failed: " . mysqli_connect_error());
        }
        $sql = "SELECT id, firstname, lastname FROM tbl_student";
        $result = mysqli_query($conn, $sql);
        if (mysqli_num_rows($result) > 0) {
   // output data of each row
        while($row = mysqli_fetch_assoc($result)) {
                echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. "
        " .$row["lastname"]."<br>";
        }
        } else {
```

```
        echo "0 results";
        }
mysqli_close($conn);
?>
```

Note: The function ***num_rows()*** checks if there are more than zero rows returned. If there are more than zero rows returned, the function ***fetch_assoc()*** puts all the results into an associative array that we can loop through. The ***while()*** loop loops through the result set and outputs the data from the id, firstname and lastname columns.

[***** is used to select all data from table.(SELECT * FROM tbl_student)]

MySQL Update Data

The UPDATE statement is used to update existing records in a table:

Example: The following examples update the record with id=2 in the "tbl_student" table:

```
<?php
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = " db_student ";
// Create connection
        $conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
        if (!$conn) {
                die("Connection failed: " . mysqli_connect_error());
        }
                $sql = "UPDATE tbl_student SET lastname='Ram' WHERE id=2";
        if (mysqli_query($conn, $sql)) {
                echo "Record updated successfully";
        } else {
                 echo "Error updating record: " . mysqli_error($conn);
        }
mysqli_close($conn);
?>
```

Note: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause , all record will be updated.

MySQL Delete Data

The DELETE statement is used to delete records from a table:

Example: The following examples delete the record with id=3 in the "tbl_student" table:

```
<?php
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "db_student";
// Create connection
        $conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
        if (!$conn) {
                die("Connection failed: " . mysqli_connect_error());
        }
// sql to delete a record
        $sql = "DELETE FROM tbl_student  WHERE id=3";
        if (mysqli_query($conn, $sql)) {
                echo "Record deleted successfully";
        } else {
                echo "Error deleting record: " . mysqli_error($conn);
        }
mysqli_close($conn);
?>
```

MySQL Insert Multiple Data

Multiple SQL statements must be executed with the *mysqli_multi_query* function.

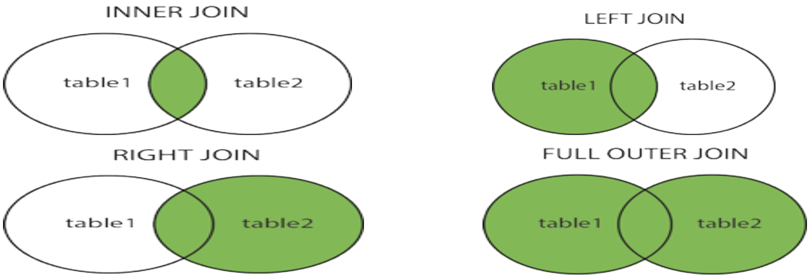
Example: The following examples add three new records to the "tbl_student" table:

```
<?php
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "db_student";
// Create connection
        $conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
        if (!$conn) {
                die("Connection failed: " . mysqli_connect_error());
        }
        $sql = "INSERT INTO tbl_student (firstname, lastname, email)
                VALUES ('John', 'Doe', 'john@example.com');";
        $sql .= "INSERT INTO tbl_student (firstname, lastname, email)
                VALUES ('Mary', 'Moe', 'mary@example.com');";
        $sql .= "INSERT INTO tbl_student (firstname, lastname, email)
                VALUES ('Julie', 'Dooley', 'julie@example.com')";
        if (mysqli_multi_query($conn, $sql)) {
                echo "New records created successfully";
        } else {
                echo "Error: " . $sql . "<br>" . mysqli_error($conn);
        }
mysqli_close($conn);
?>
```

SQL JOIN command

Here are the different types of the JOINs in SQL:

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table



(INNER) JOIN

The INNER JOIN keyword selects records that have matching values in both tables.

Syntax:

SELECT *column_name(s)*
FROM *table1*
INNER JOIN *table2*
ON *table1.column_name* = *table2.column_name*;

SQL Statement:

SELECT Orders.OrderID, Customers.CustomerName FROM Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

Syntax:

SELECT *column_name(s)*
FROM *table1*
LEFT JOIN *table2*
ON *table1.column_name* = *table2.column_name*;

SQL Statement:

SELECT Orders.OrderID, Customers.CustomerName FROM Orders LEFT JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

Syntax:

SELECT *column_name(s)*
FROM *table1*
RIGHT JOIN *table2*
ON *table1.column_name* = *table2.column_name*;

SQL Statement:

SELECT Orders.OrderID, Customers.CustomerName FROM Orders RIGHT JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

FULL JOIN

The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

Syntax:

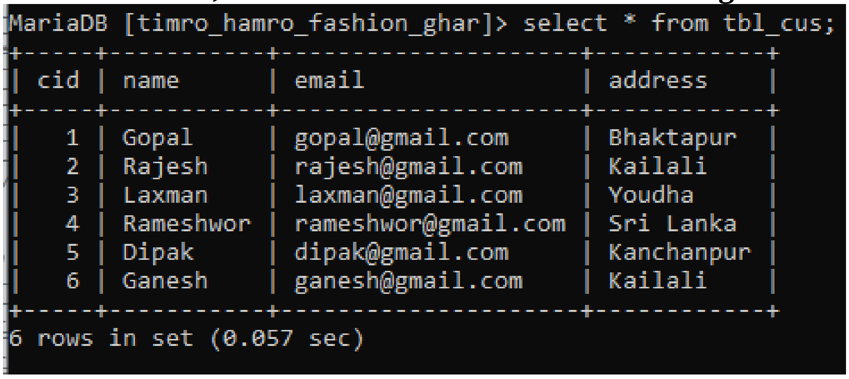
SELECT *column_name(s)*
FROM *table1*
FULL JOIN *table2*
ON *table1.column_name* = *table2.column_name*;

SQL Statement:

SELECT Orders.OrderID, Customers.CustomerName FROM Orders FULL JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

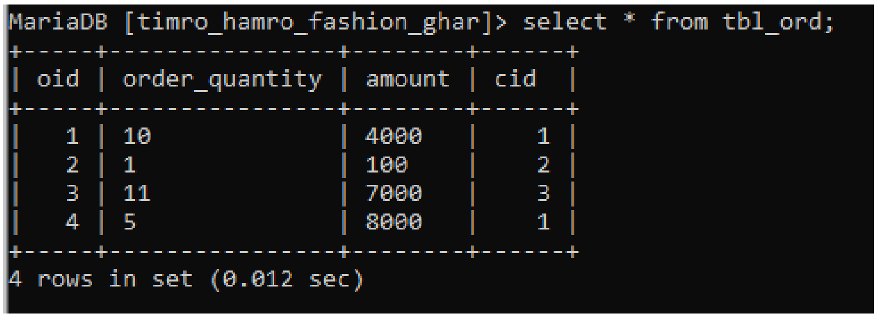
(In MySql CROSS JOIN instead of FULL JOIN)

Let , we have two tables , customers and orders as shown in figure below:



```
MariaDB [timro_hamro_fashion_ghar]> select * from tbl_cus;
+-----+-----------+---------------------+------------+
| cid | name      | email               | address    |
+-----+-----------+---------------------+------------+
|   1 | Gopal     | gopal@gmail.com     | Bhaktapur  |
|   2 | Rajesh    | rajesh@gmail.com    | Kailali    |
|   3 | Laxman    | laxman@gmail.com    | Youdha     |
|   4 | Rameshwor | rameshwor@gmail.com | Sri Lanka  |
|   5 | Dipak     | dipak@gmail.com     | Kanchanpur |
|   6 | Ganesh    | ganesh@gmail.com    | Kailali    |
+-----+-----------+---------------------+------------+
6 rows in set (0.057 sec)
```

figure: tbl_cus table



```
MariaDB [timro_hamro_fashion_ghar]> select * from tbl_ord;
+-----+----------------+--------+------+
| oid | order_quantity | amount | cid  |
+-----+----------------+--------+------+
|   1 | 10             | 4000   |    1 |
|   2 | 1              | 100    |    2 |
|   3 | 11             | 7000   |    3 |
|   4 | 5              | 8000   |    1 |
+-----+----------------+--------+------+
4 rows in set (0.012 sec)
```

figure: tbl_ord table

Source Code(example):

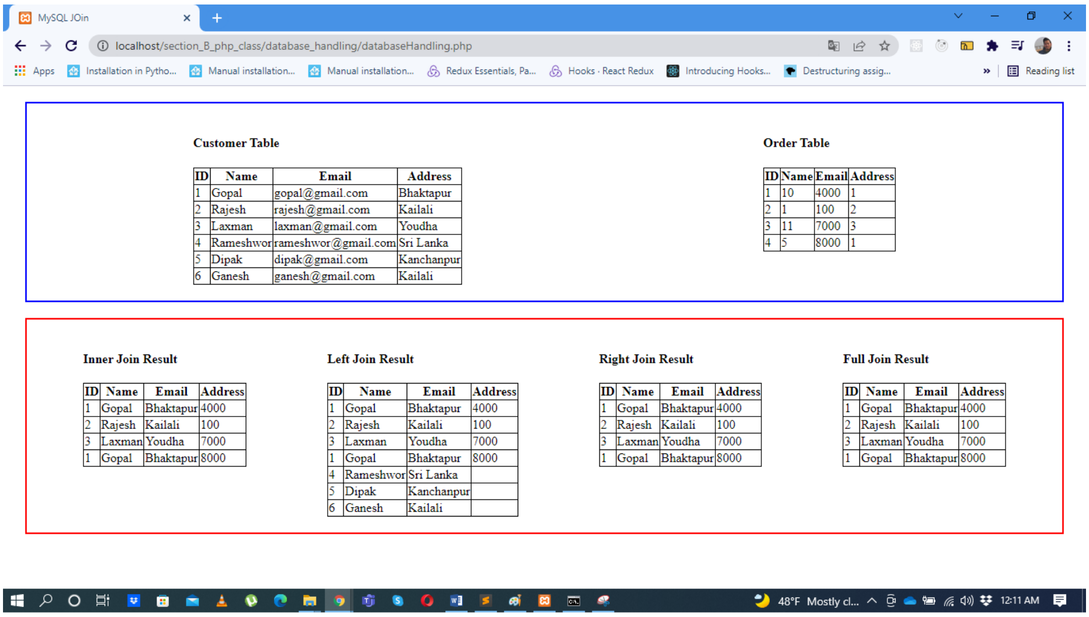
```
<!-- databaseHandling.php -->
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>MySQL JOin</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <?php include_once('query.php');
		if(!$conn){
			die("Database not connected.");
		}
```

```
 ?>
 <div>
		<div id="orinal_table">
			<div id="customer_tbl">
				<h4>Customer Table</h4>
				<?php include_once('customer.php') ?>
			</div>
			<div id="order_tbl">
				<h4>Order Table</h4>
				<?php include_once('order.php'); ?>
			</div>
		</div>
		<div id="after_join">
			<div id="inner_join">
				<h4>Inner Join Result</h4>
				<?php include_once('inner_join.php') ?>
			</div>
			<div id="left_join">
				<h4>Left Join Result</h4>
				<?php include_once('left_join.php'); ?>
			</div>
			<div id="right_join">
				<h4>Right Join Result</h4>
				<?php include_once('right_join.php'); ?>
			</div>
			<div id="full_join">
				<h4>Full Join Result</h4>
				<?php include_once('full_join.php'); ?>
			</div>
		</div>
 </div>
</body>
</html>
<!-- fileName: query.php -->
<?php
 $user = 'root';
 $host = 'localhost';
 $pass = '';
 $db = 'timro_hamro_fashion_ghar';
 //connection to database
		$conn = mysqli_connect($host,$user,$pass,$db);
 //query to create customer table
		$queryToCreateCustomerTable = "CREATE TABLE IF NOT EXISTS tbl_cus(cid INT UNSIGNED
AUTO_INCREMENT PRIMARY KEY, name VARCHAR(20),email VARCHAR(40) NOT NULL, address
VARCHAR(40) NOT NULL,phone VARCHAR(30))";
 //query to create order table (using foreign key)
		$queryToCreateOrderTable = "CREATE TABLE IF NOT EXISTS tbl_ord(oid INT UNSIGNED
AUTO_INCREMENT,order_date date,amount INT,cid INT UNSIGNED,PRIMARY KEY(oid),FOREIGN
KEY(cid) REFERENCES customer_tbl(cid))";
 //query to retrive information from customer table
		$sqlForCustomer = "SELECT * FROM tbl_cus";
 //query to retrive information from order table
		$sqlForOrder = "SELECT * FROM tbl_ord";

 //query for inner join
```

```
        $sqlForInnerJoin = "SELECT tbl_cus.cid,name,address,amount FROM tbl_cus JOIN tbl_ord
ON tbl_cus.cid = tbl_ord.cid";

  //query for left join()
        $sqlForLeftJoin = "SELECT tbl_cus.cid,name,address,amount FROM tbl_cus LEFT JOIN
tbl_ord ON tbl_cus.cid = tbl_ord.cid";

  //query for right join()
        $sqlForRightJoin = "SELECT tbl_cus.cid,name,address,amount FROM tbl_cus RIGHT JOIN
tbl_ord ON tbl_cus.cid = tbl_ord.cid";

  //query for full join()
        $sqlForFullJoin = "SELECT tbl_cus.cid,name,address,amount FROM tbl_cus CROSS JOIN
tbl_ord ON tbl_cus.cid = tbl_ord.cid";
?>
<!-- customer.php -->
<?php
  include('query.php');
  $cusResult = mysqli_query($conn,$sqlForCustomer);
  if(mysqli_num_rows($cusResult) > 0){
        echo "<table>";
                echo "<tr>";
                        echo "<th>ID</th>";
                        echo "<th>Name</th>";
                        echo "<th>Email</th>";
                        echo "<th>Address</th>";
                echo "</tr>";
                        while($rows = mysqli_fetch_assoc($cusResult)){
                                echo "<tr>";
                                        echo "<td>".$rows['cid']."</td>";
                                        echo "<td>".$rows['name']."</td>";
                                        echo "<td>".$rows['email']."</td>";
                                        echo "<td>".$rows['address']."</td>";
                                echo "</tr>";
                        }
        echo "</table>";
  }
?>
<!-- order.php -->
<?php
  include('query.php');
  $ordResult = mysqli_query($conn,$sqlForOrder);
  if(mysqli_num_rows($ordResult) > 0){
        echo "<table>";
                echo "<tr>";
                        echo "<th>ID</th>";
                        echo "<th>Name</th>";
                        echo "<th>Email</th>";
                        echo "<th>Address</th>";
                echo "</tr>";
                        while($rows = mysqli_fetch_assoc($ordResult)){
                                echo "<tr>";
                                        echo "<td>".$rows['oid']."</td>";
                                        echo "<td>".$rows['order_quantity']."</td>";
                                        echo "<td>".$rows['amount']."</td>";
```

```
                     echo "<td>".$rows['cid']."</td>";
                 echo "</tr>";
             }
     echo "</table>";
 }
?>
<!-- inner_join.php -->
<?php
 include('query.php');
 $innerResult = mysqli_query($conn,$sqlForInnerJoin);
 if(mysqli_num_rows($innerResult) > 0){
     echo "<table>";
         echo "<tr>";
             echo "<th>ID</th>";
             echo "<th>Name</th>";
             echo "<th>Email</th>";
             echo "<th>Address</th>";
         echo "</tr>";
             while($rows = mysqli_fetch_assoc($innerResult)){
                 echo "<tr>";
                     echo "<td>".$rows['cid']."</td>";
                     echo "<td>".$rows['name']."</td>";
                     echo "<td>".$rows['address']."</td>";
                     echo "<td>".$rows['amount']."</td>";
                 echo "</tr>";
             }
     echo "</table>";
 }
?>
<!-- left-join.php -->
<?php
 include('query.php');
 $leftResult = mysqli_query($conn,$sqlForLeftJoin);
 if(mysqli_num_rows($leftResult) > 0){
     echo "<table>";
         echo "<tr>";
             echo "<th>ID</th>";
             echo "<th>Name</th>";
             echo "<th>Email</th>";
             echo "<th>Address</th>";
         echo "</tr>";
             while($rows = mysqli_fetch_assoc($leftResult)){
                 echo "<tr>";
                     echo "<td>".$rows['cid']."</td>";
                     echo "<td>".$rows['name']."</td>";
                     echo "<td>".$rows['address']."</td>";
                     echo "<td>".$rows['amount']."</td>";
                 echo "</tr>";
             }
     echo "</table>";
 }
?>
<!-- right_join.php -->
<?php
 include('query.php');
 $rightResult = mysqli_query($conn,$sqlForRightJoin);
 if(mysqli_num_rows($rightResult) > 0){
        echo "<table>";
                echo "<tr>";
                        echo "<th>ID</th>";
                        echo "<th>Name</th>";
                        echo "<th>Email</th>";
                        echo "<th>Address</th>";
                echo "</tr>";
                        while($rows = mysqli_fetch_assoc($rightResult)){
                                echo "<tr>";
                                        echo "<td>".$rows['cid']."</td>";
                                        echo "<td>".$rows['name']."</td>";
                                        echo "<td>".$rows['address']."</td>";
                                        echo "<td>".$rows['amount']."</td>";
                                echo "</tr>";
                        }
        echo "</table>";
 }
?>
<!-- full_join.php -->
<?php
 include('query.php');
 $fullResult = mysqli_query($conn,$sqlForFullJoin);
 if(mysqli_num_rows($fullResult) > 0){
        echo "<table>";
                echo "<tr>";
                        echo "<th>ID</th>";
                        echo "<th>Name</th>";
                        echo "<th>Email</th>";
                        echo "<th>Address</th>";
                echo "</tr>";
                        while($rows = mysqli_fetch_assoc($fullResult)){
                                echo "<tr>";
                                        echo "<td>".$rows['cid']."</td>";
                                        echo "<td>".$rows['name']."</td>";
                                        echo "<td>".$rows['address']."</td>";
                                        echo "<td>".$rows['amount']."</td>";
                                echo "</tr>";
                        }
        echo "</table>";
 }
?>
```

Output:



Customer Table

ID	Name	Email	Address
1	Gopal	gopal@gmail.com	Bhaktapur
2	Rajesh	rajesh@gmail.com	Kailali
3	Laxman	laxman@gmail.com	Youdha
4	Rameshwor	rameshwor@gmail.com	Sri Lanka
5	Dipak	dipak@gmail.com	Kanchanpur
6	Ganesh	ganesh@gmail.com	Kailali

Order Table

ID	Name	Email	Address
1	10	4000	1
2	1	100	2
3	11	7000	3
4	5	8000	1

Inner Join Result

ID	Name	Email	Address
1	Gopal	Bhaktapur	4000
2	Rajesh	Kailali	100
3	Laxman	Youdha	7000
1	Gopal	Bhaktapur	8000

Left Join Result

ID	Name	Email	Address
1	Gopal	Bhaktapur	4000
2	Rajesh	Kailali	100
3	Laxman	Youdha	7000
1	Gopal	Bhaktapur	8000
4	Rameshwor	Sri Lanka	
5	Dipak	Kanchanpur	
6	Ganesh	Kailali	

Right Join Result

ID	Name	Email	Address
1	Gopal	Bhaktapur	4000
2	Rajesh	Kailali	100
3	Laxman	Youdha	7000
1	Gopal	Bhaktapur	8000

Full Join Result

ID	Name	Email	Address
1	Gopal	Bhaktapur	4000
2	Rajesh	Kailali	100
3	Laxman	Youdha	7000
1	Gopal	Bhaktapur	8000

GET and POST

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. Both GET and POST are treated as $_GET and $_POST. Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character. $_GET is an array of variables passed to the current script via the URL parameters. Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. GET may be used for sending non-sensitive data.

Note: *GET* should never be used for sending passwords or other sensitive information!

Example:

```
<?php
  if($_GET['name'] || $_GET['age']) {
    echo "Welcome ". $_GET['name']. "<br />";
    echo "You are ". $_GET['age']. " years old.";
    exit();
  }
?>
<html>
  <body>
    <form action = "<?php $_PHP_SELF ?>" method = "GET">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" value="submit" />
    </form>
  </body>
</html>
```

The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING. $_POST is an array of variables passed to the current script via the HTTP POST method. Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send. Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

Example:

```
<?php
  if( $_POST["name"] || $_POST["age"] ) {
    if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
      die ("invalid name and name should be alpha");
    }
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";
    exit();
  }
?>
<html>
  <body>
    <form action = "<?php $_PHP_SELF ?>" method = "POST">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>
  </body>
</html>
```

The $_REQUEST variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE. The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

Example:

```
<?php
  if( $_REQUEST["name"] || $_REQUEST["age"] ) {
    echo "Welcome ". $_REQUEST['name']. "<br />";
    echo "You are ". $_REQUEST['age']. " years old.";
    exit();
  }
?>
<html>
  <body>
    <form action = "<?php $_PHP_SELF ?>" method = "POST">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>
  </body>
</html>
```

Pattern Matching

Regular expressions are nothing more than a sequence or pattern of characters itself. They provide the foundation for pattern-matching functionality. Using regular expression you can search a particular string inside a another string, you can replace one string by another string and you can split a string into many chunks. About Regular Expression , discussed in **chapter 4**, here we only discuss on some functions that are used for pattern matching.

These expression often used with SQL(Structured Query Language) to create, modify and manage data in databases and data-driven application.

Some examples:

^**[0-9][a-z]$** = match any string that DOES begin with a number and has a
lowercase letters.

^**[^0-9][A-Z]$** = match any string that DOES NOT begin with a number and has a
Capital letters.

^**Z{1,3}$** = matches Z,ZZ and ZZZ

^**Z{3,5}$** = matches ZZZ,ZZZZ and ZZZZZ

^\-**?[0-9]{0,}\.?[0-9]{0,}$** = In this expression everything is between the ^ and **$,** so we are looking for an exact pattern match from beginning to end . The first part of the expression, "\-**?**" means look for a minus sign and uses the \ since it means a literal . The **?** means that the minus sign is optional. The second part of the expression is "**[0-9]{0,}**" and means look for numbers with **[0-9]** and find 0 or more of them with **{0,}**. Other part are also similar where "." is optional.

^**[0-9]{1,}$** = match any string that begins with one or more numbers. To say this +

^**.+@.+\..+$** = match any string that begins with one or more non-new-line characters, has an "**@**" , is then followed by one or more other non-new-line characters, then has a literal "**.**" and other non-new-line characters.

^**(L|Th)ink$** = matches either **Link** OR **Think**

Some PHP's Regexp PERL Compatible Functions

preg_match() Function:

The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.

Example:

```
<?php
        $a="hello world";
         if(preg_match('/hi/',$a))
         {
```

```
                    echo "Pattern found";
            }
             else{
                    echo "Pattern not found";
            }
 ?>
```

Example:

```
<?php
            $email="santoshbist69@gmail.com";

            if(preg_match("/^.+@.+\..+/",$email)==true)
            {
                    echo("that's true id");
            }
             else{
                     echo("that's fake id");
            }
 ?>
```

preg_replace() function

The preg_replace() function operates just like POSIX function ereg_replace(), except that regular expressions can be used in the pattern and replacement input parameters.

Example:

```
<?php
   $copy_date = "Copyright 1999";
   $copy_date = preg_replace("([0-9]+)", "2000", $copy_date);
   print $copy_date;
?>
```

PHP preg_match_all() Function

The preg_match_all() function returns the number of matches of a pattern that were found in a string and populates a variable with the matches that were found.

Syntax:

preg_match_all(pattern, input, matches, flags, offset)

pattern – Regular expression. Required.
input – String in which search is performed. Required.
matches – Array containing all matches. Optional.
flag – Optional. A set of options that change how the matches array is structured. PREG_PATTERN_ORDER(default), PREG_SET_ORDER, PREG_OFFSET_CAPTURE, PREG_UNMATCHED_AS_NULL.
offset – Defaults to 0. Optional. Indicates how far into the string to begin searching.

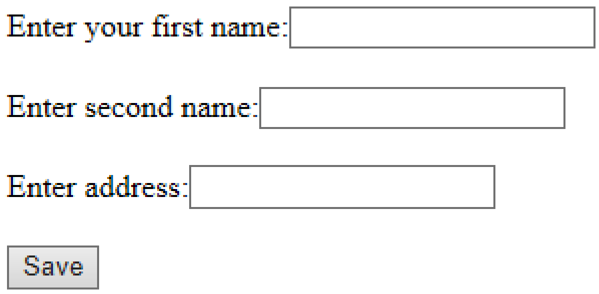
Example: Validation of email in php

```
<!-- reglarExpression.php -->
<!DOCTYPE html>
<html>
<head>
            <meta charset="utf-8">
            <meta name="viewport" content="width=device-width, initial-scale=1">
            <title>Email Validation</title>
```

```
</head>
<body>
        <form action="response.php" method="post">
                Email: <input type="email" name="email" required /></br></br>
                <input type="submit" name="btn" value="Email Validation" />
        </form>
        <span id="v-result"></span>

</body>
</html>
<!-- response.php -->
<?php
        $emailPattern = '/^[^-9][a-z0-9._]{3,}@[a-z]{2,}.{1}[a-z]{2,}(.[a-z]+)?$/';
        $email = $_POST['email'];
        if (preg_match_all($emailPattern, $email)) {
                // code...
                echo "Valid";
        }else{
                echo "Not valid.";
        }
?>
```

Some Questions:

Q1. Write necessary php scripts to make connection with the database and insert/save form data into database . Form given below:

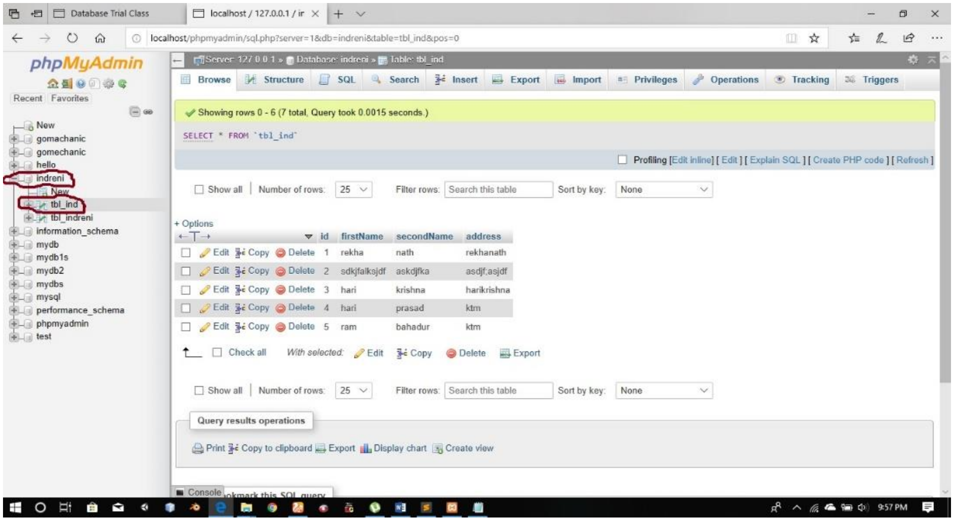


Answer:

```
<!DOCTYPE html>
<html>
<head>
  <title>Database Trial Class</title>
</head>
<body>
  <form method="post">
    Enter your first name:<input type="text" name="fname"><br><br>
    Enter second name:<input type="text" name="lname"><br>
    <br>
    Enter address:<input type="text" name="address"><br><br>
    <input type="submit" name="imply" value="Save">
  </form>
</body>
</html>
<?php
         if(isset($_POST['imply']))
```

```
        {
        $fname=$_POST["fname"];
        $lname=$_POST["lname"];
        $address=$_POST["address"];

        $host='localhost';
        $user='root';
        $pass='';
        $db='indreni';

        $con=mysqli_connect($host,$user,$pass,$db);

        if($con)
                echo 'database connected successfully';
        echo'<br>';
                $sql="insert into tbl_ind(firstName,secondName,address) values
        ('".$fname."',        '".$lname."','".$address."')";
        $query=mysqli_query($con,$sql);
        if($query)
                echo 'Data is inserted';
        }
?>
```

Output: Data will be inserted when clicked on Save button. Here database base name is ***indreni*** and table name is ***tbl_ind.***



Showing rows 0 - 6 (7 total, Query took 0.0015 seconds.)

```
SELECT * FROM `tbl_ind`
```

□ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

	id	firstName	secondName	address
□ Edit Copy Delete	1	rekha	nath	rekhanath
□ Edit Copy Delete	2	sdkjfalksjdf	askdjfka	asdjf,asjdf
□ Edit Copy Delete	3	hari	krishna	harikrishna
□ Edit Copy Delete	4	hari	prasad	ktm
□ Edit Copy Delete	5	ram	bahadur	ktm

Q2. Write JavaScript code to find the factorial of a number requesting a number from user prompt.

Answer:

```
<!doctype html>
<html>
<head>
<script>
        function show(){
        var i, numb, fact;
        fact=1;
        numb=prompt("Enter the number","");
        for(i=1; i<=numb; i++)
        {
                fact= fact*i;
        }
        document.write("The factorial of "+numb+" is ",fact);
        }
        show();//function calling
</script>
</head>
<body>
<p>Calculating Factorial of number through prompt.</p>
</body>
</html>
```

OR if not using Prompt

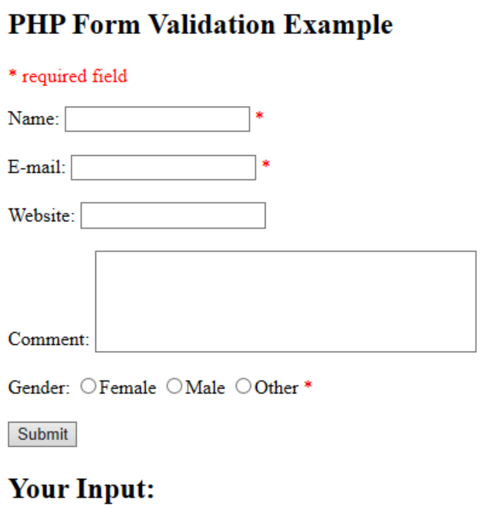
```
<!doctype html>
<html>
<head>
<script>
        function show(){
        var i, numb, fact;
        fact=1;
        numb=Number(document.getElementById("num").value);
        for(i=1; i<=numb; i++)
        {
                fact= fact*i;
        }
        document.getElementById("answer").value= fact;
        }
</script>
</head>
<body>
Enter Num: <input id="num">
<button onclick="show()">Factorial</button><br><br>
<p>The factorial of a given number is <input id="answer">
</body>
</html>
```

Q3. Validation of number and character:

Answer:

```
<!DOCTYPE html>
<html>
<head>
        <title>validate number and characters</title>
</head>
<script>
//this function is used to validate character input
        function myFunction() {
//for character input
        var correct=/^[A-Za-z]+$/;
        var b=document.getElementById("user_name").value;
        if (b=="") {
                document.getElementById("msg").innerHTML="**Must fill";
                return false;
        }
        if (b.length<3) {
                document.getElementById("msg").innerHTML="**Character must be
        greater than 3";
                return false;
        }
        if (b.length>20) {
                document.getElementById("msg").innerHTML="**Character less than
        25";
                return false;
        }
        if (b.match(correct))
                true;
        else
        {
                document.getElementById("msg").innerHTML="**Only character
        allows";
        return false;
}
}
//This function is used to validate number input
        function myFun(){
        var x, text;
// Get the value of the input field with id="numb"
        x = document.getElementById("numb").value;
// If x is Not a Number or less than one or greater than 10
        if (isNaN(x) || x < 1 || x > 10) {
                text = "Input not valid";
        } else {
                text = "Input OK";
        }
                document.getElementById("demo").innerHTML = text;
        }
</script>
<body>
        <p>Please input a number between 1 and 10:</p>
        <input id="numb">
        <button type="button" onclick="myFun()">click me</button>
        <p id="demo"></p>
        <form onsubmit="return myFunction()">
                user name:<input type="text" id="user_name">
                <span id="msg"></span>
                <input type="submit" value="submit">
        </form>
</body>
</html>
```

Q4. Validate the Form below.



PHP Form Validation Example

* required field

Name: ______ *

E-mail: ______ *

Website: ______

Comment: ______

Gender: ○Female ○Male ○Other *

Submit

Your Input:

Answer:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid
    if                      (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>
```

Q5. How to reverse String using JavaScript?

Answer:

```
<!DOCTYPE html>

<html>

<head>

 <title>Reverse given string from user input!</title>

</head>

<script type="text/javascript">

 function functionReverse(){

         var reverseString="";

         var inputString;

 inputString=document.getElementById('getVal').value;

         for(var i=inputString.length-1;i>=0;i--){

                 reverseString+=inputString[i];

         }

         document.getElementById('section1').innerHTML=reverseString;


 }

 function function_Split_Reverse_Join(){

         var inputString;

         inputString=document.getElementById('getVal').value;

         //split-method :-split string as an array where blank space is.

         //reverse -method:- reverse splited string

         //join-method:- join all string having blank space

         var word=inputString.split(" ").reverse().join(" ");

         document.getElementById('section2').innerHTML=word;
```

```
  }


  function functionComment(){
                        var str="Hope you <b>understand</b> how to reverse string in javascript. <b>If
not</b> then read/analysis and run program carefully. ";
                        document.getElementById('section3').innerHTML=str+"<b>Thank you!</b>";
  }
</script>


<body>
  Input:<input type="text" id="getVal" placeholder="Input String you want">
  <br><br>
  <input type="button" value="Reverse" onclick="functionReverse()">
  <input type="button" value="Split-Reverse-Join" onclick="function_Split_Reverse_Join()">
  <input type="button" value="Comment" onclick="functionComment()">
  <br><br>
            <p id="section1"></p>
            <p id="section2"></p>
            <p id="section3"></p>
            <p id="section4"></p>
</body>
</html>
```

Assignment: *Explain all GLOBAL VARIABLES in PHP*