

IT1100 - Internet and Web Technologies

Introduction

Internet and Web technologies

- Module Code IT1100
- Credit Points 04

Method of Delivery

- 2 hours - lectures
 - 1 hour - tutorials
 - 2 hours - labs
-
- Enrollment Key IT1100

Assessment Criteria

Component	%
Mid Semester Exam	20%
Assignment – part 01	10%
Assignment – part 02	20%
Final Exam	50%

Important

To pass this module Student need to obtain a pass mark in both “Continues assessment” and “End of the Semester Examination “ components which would result in an overall mark that would qualify for a C grade or above.

Assignments

You need to get into **5-member** group for the assignment within the same subgroup.

- You must “Develop a web Application”
- Project Titles are given by us

Assignment Submissions

- 5th Week – Documentation of your project plan .
- 12th Week – Final project submission
- 12th and 13th Week - Viva.

Reference Materials



- W3 school - <https://www.w3schools.com/>
- J. Reynolds and R. Mofazali, *The complete e-commerce book: design, build, and maintain a successful web-based business*, 1st. ed., C M P Books, 2000.
- R. Nixon, *Learning PHP, MySQL, JavaScript and CSS: A step-by-step guide to creating dynamic websites*, O'Reilly Media, Inc., 2012.
- H. Sharp, Y. Rogers, and J. Preece, *Interaction Design: Beyond Human-Computer Interaction*, 2nd ed. Wiley, 2007.
- Tutorial point - tutorialspoint.com

PLEASE CHECK THE
COURSE WEB
REGULARLY



Concepts and technologies Associated with the Web applications

Lecture 01

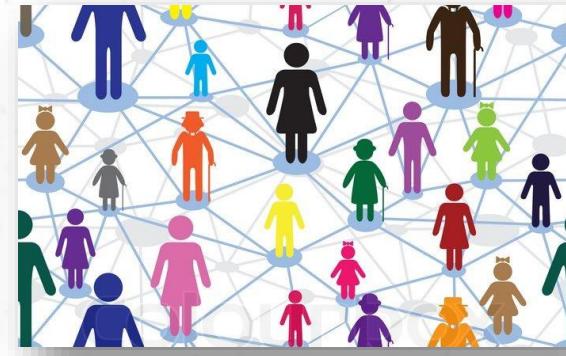
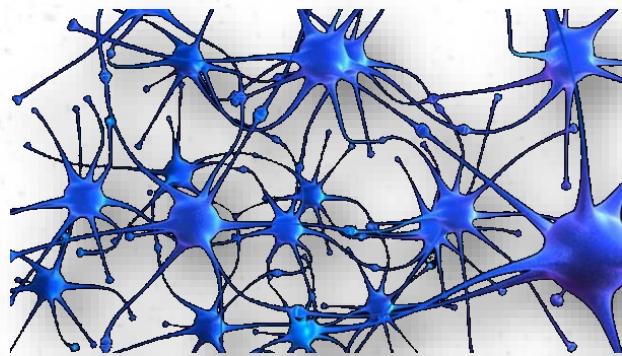
Content

- Data networks and the Internet
- Network Services and Protocols
- Web server and the Browser
- Markup languages

Data Networks and Internet

What is a network?

- A **network** is (according to the Cambridge Dictionary) a **large system** consisting of **many similar parts** that are **connected together** to allow **movement** or **communication** along the parts, or between the parts and a control centre.



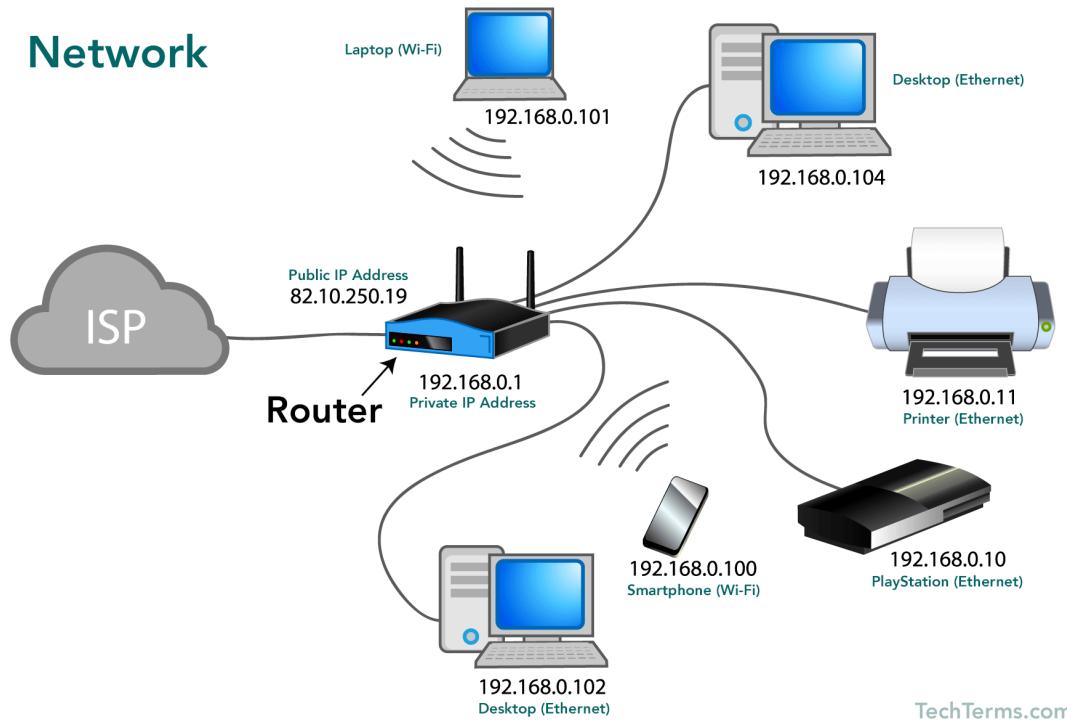
Different types of networks

- There are different types of networks available (according to the nature of the usage)
 - Telecommunication networks
 - Television or radio network
 - Transport networks
 - Social networks
 - **Computer or data networks**



Computer and Data Network

Network



- A computer network, or data network is
 - a **digital** telecommunications network, which allows **nodes** to share **resources**.
 - In computer networks, **computing devices** exchange data with each other using connections between nodes (**data links**).
 - These **data links** are established over **cable media** such as wires or optic cables, or **wireless media** such as WiFi.

Application of Data Networks

■ Resource Sharing

- Hardware (computing resources, disks, printers)
- Software (application software)

⋮ ⋮
⋮ ⋮
⋮ ⋮

■ Information Sharing

- Easy accessibility from anywhere (files, databases)
- Search Capability (WWW)

■ Communication

- Email Message
- broadcast

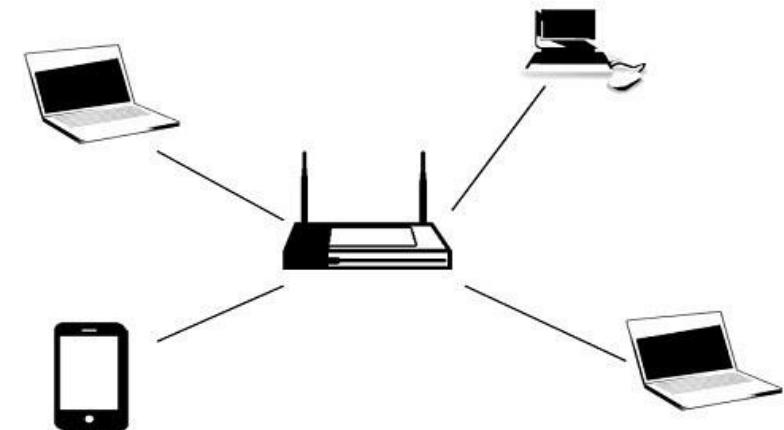
⋮ ⋮ ⋮ ⋮

■ Remote computing

Types of data networks - LAN

Local Area Network – LAN

- Network in small geographical Area (Roo Building or a Campus) is called LAN (Loca Area Network)
- Local Area Networks are **privately-owned** networks within a small area, usually a si building or campus of up to a **few kilome**
- Since it is restricted in size, that means their data transmission time can be known in advance, and the network management would be easier.

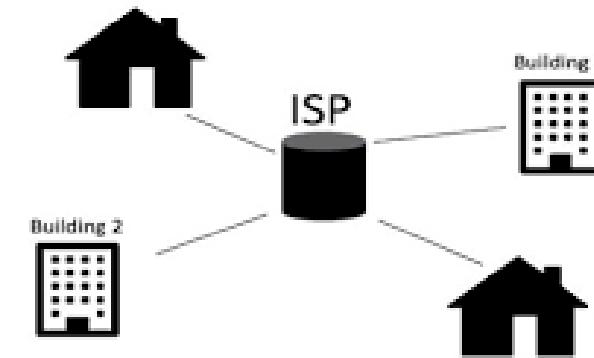


Source: <http://cityinfrastructure.com/Data/Daa.html>

Types of data networks - MAN

Metropolitan Area Network – MAN

- A Metropolitan Area Network (MAN) is a network that is utilized across multiple buildings
- Commonly used in schools, campuses ,hospitals , banks or large companies with multiple buildings
- Is larger than a LAN, but smaller than a WAN
- Is also used to mean the interconnection of several LANs by bridging them together. This sort of network is also referred to as a campus network

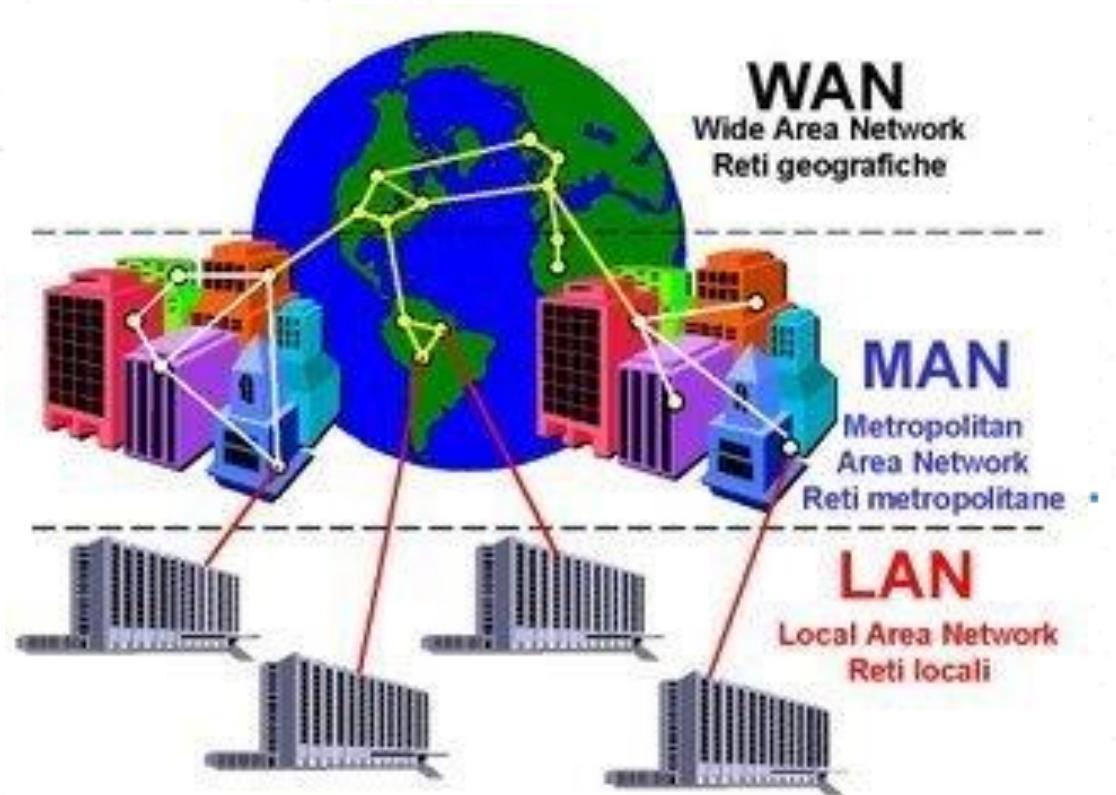


Source: <http://cityinfrastructure.com/Data/Daa.html>

Types of data networks - WAN

Wide Area Network – WAN

- A Wide Area Network is a network spanning a large geographical area of around several hundred miles to across the globe
- May be privately owned or leased
- Also called “enterprise networks” if they are privately owned by a large company
- Can be connected through cable, fiber or satellite
- Is typically slower and less reliable than a LAN



Internet

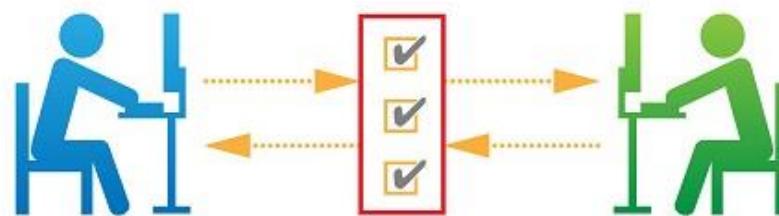
- The Internet is the **global system of interconnected computer networks** that use the Internet **protocol** suite to link devices worldwide.
- It is a **network of networks**
- Consists of private, public, academic, business, and government networks of local to **global scope**.
- Linked by a broad array of electronic, wireless, and optical networking technologies.

Source <https://en.wikipedia.org/wiki/Internet>

Network Services and Protocols

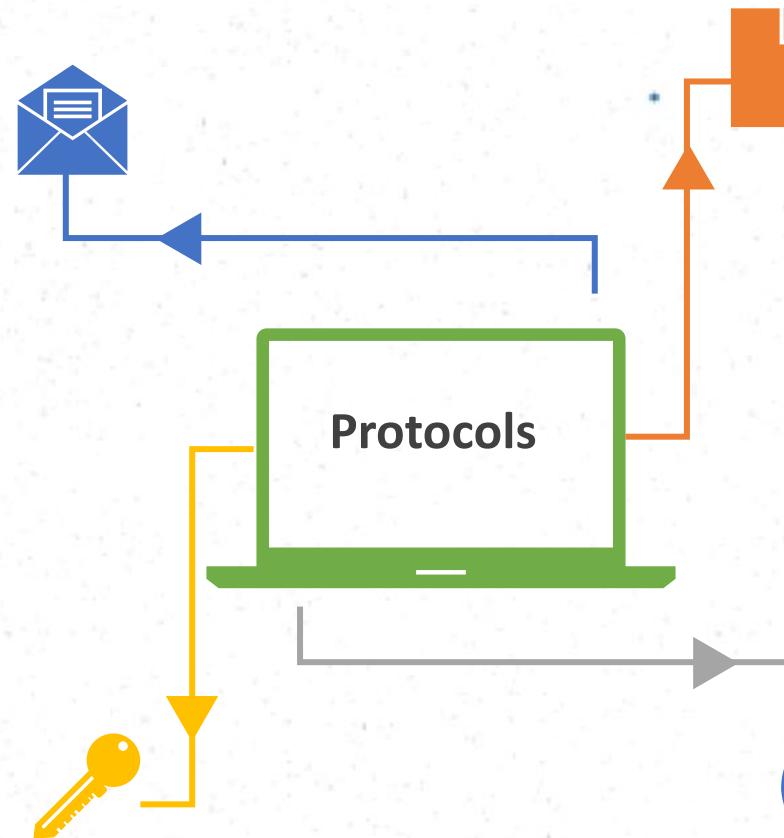
Protocols

- A protocol is a
 - system of rules that allow two or more entities of a communications system to transmit information (wiki)
 - **the formal system of rules for correct behavior on official occasions (Cambridge)**



Different Types of Services and Protocols

Mail service (POP3/SMTP/IMAP)



File Transfer (FTP)

Remote Logging (SSH)

Web (HTTP/HTTPS)

Web server and the Browser

Identify the browsers

We Use a browser to send HTTP/ HTTPS request



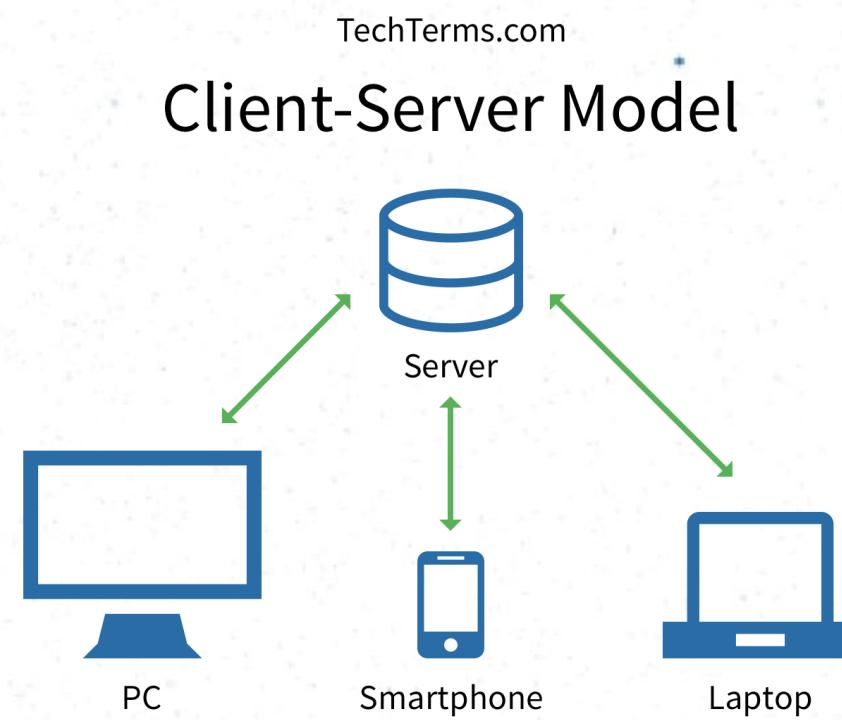
The browser

- Usually, the clients use the web browser to access the web application in the server, based on the request-response pattern.
 1. The user enters the address of the web server (domain name) into the browser.
 2. The browser sends a request to the web server
 3. The server responses with the client components
 4. The client components are loaded into the browser
 5. The browser reads the content and renders

The Server

- A server is a software, which knows how to handle the requests and responses, while providing a specific service
- A web server is used to host a web application.
 - **Apache (for php development)**
 - **Tomcat (for JAVA development)**
 - **IIS (for .NET/ASP development)**
- Web server knows how to communicate with the clients using the **HTTP/HTTPS**

Client and Server



Types of languages

- High level/Compiled languages – Java, C, C++
⋮ ⋮ ⋮
- Scripting languages – JS, PHP, Python
⋮ ⋮ ⋮
- Markup languages – XML, HTML, XHTML
⋮ ⋮ ⋮

Markup Languages

extensible Markup Language

- Designed to store and transport data
- Both human- and machine-readable (self descriptive)
- Often used for distributing data over networks
- Used by many other tools like protocols

```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
       more questions later.-->
</quiz>
```

XML

XML

- The main and the only component of XML is called an **element**
- An element has 3 components
 1. Start tag
 2. Body
 3. End tag
- **No predefined set of elements, attributes, and values for attributes**

<Tag_name>IWT</Tag_name>

XML

- An element has a name
 - **Element names are case-sensitive**
 - Element names must start with a letter or underscore
 - Element names cannot start with the letters, xml (or XML, or Xml, etc)
 - Element names can contain letters, digits, hyphens, underscores, and periods
 - Element names cannot contain spaces
 - Any name can be used, no words are reserved (except xml)

<Module>IWT</Module>

XML

- Element names – naming styles

<u>Style</u>	<u>Example</u>	<u>Description</u>
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first

XML

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



- This is the XML declaration
 - Provides the instructions for the processor to understand the details of the XML file
 - Encoding attribute indicates the character set
 - UTF-8 = Unicode Transformation Format (with 8-bit blocks to represent a character)
- An element may have attribute(s)
 - Attributes describe the element
- Attribute value is always quoted (either **single** or **double** quote)
- <person id="1">Saman</person>



XML

- There can be multiple **attributes** for an element

```
<person id="1" age="35">  
    Saman  
</person>
```

- Attributes are separated by a space
 - There are special type of element with a single self closing tag
- ```
<age/>
```

# XML

- Elements can be nested

```
<person id="1">
 <firstname>Saman</firstname>
 <lastname>De Silva</lastname>
 <age/>
</person>

<person id="2">
 <firstname>Saman</firstname>
 <lastname>De Silva</lastname>
 <age>28</age>
</person>
```

- The first element, which wraps and holds the other elements is called, the root element

# XML

- Learn more about XML

<https://www.w3schools.com/xml/default.asp>

- HTML Unicode (UTF-8) Reference

[https://www.w3schools.com/charsets/ref\\_html\\_utf8.asp](https://www.w3schools.com/charsets/ref_html_utf8.asp)

# Question

- Write XML code to store following personal data
    - Name
    - Gender
    - Age
    - School



# HTML – Hyper Text Markup Language

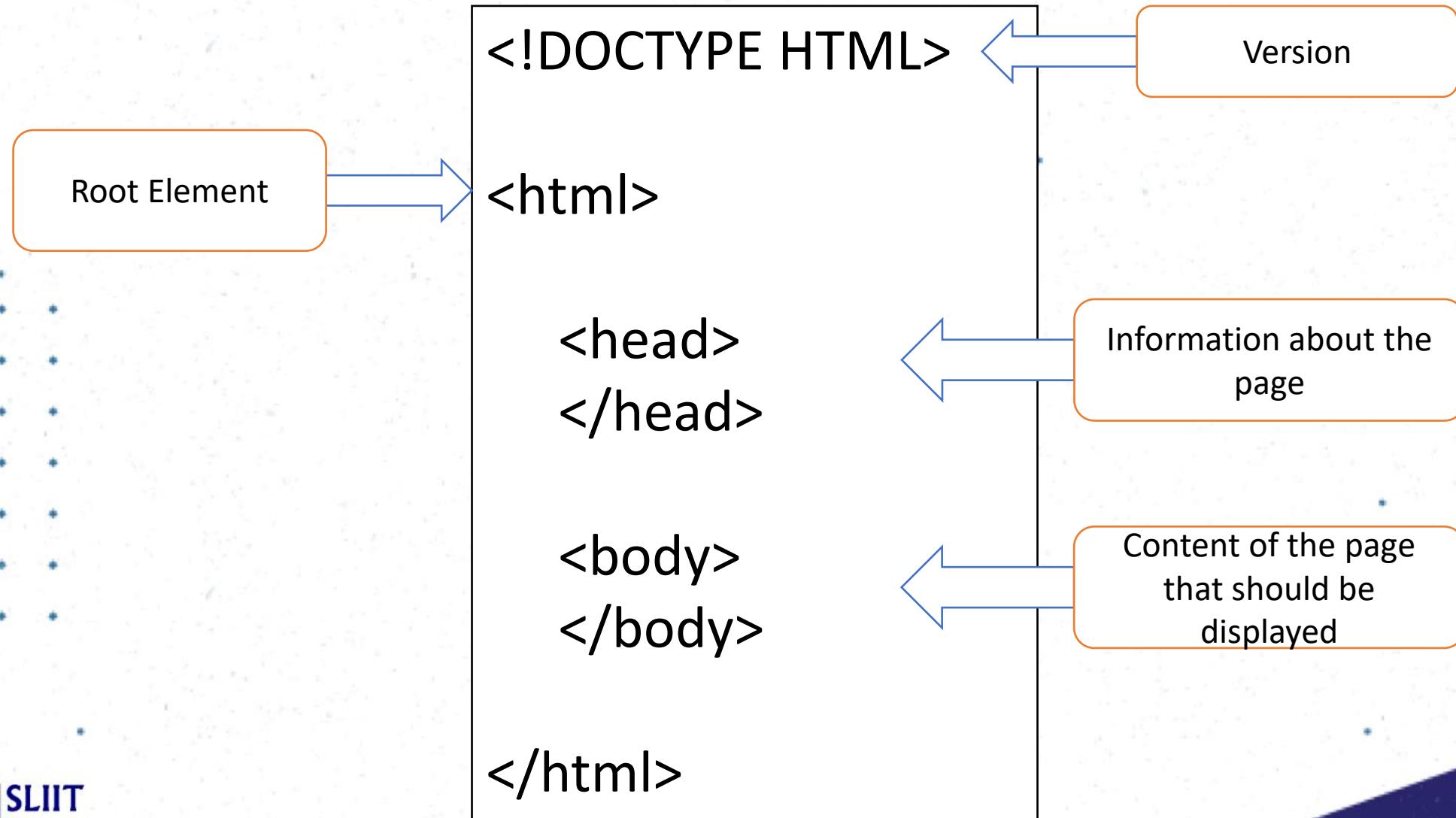


- HTML is the standard language to develop pages
- The web browser knows to read the HTML document and render the content, showing a nice GUI for web sites/applications
- HTML has a predefined set of elements, attributes, and values for some attributes

# HTML – Hyper Text Markup Language

- HTML document (or the web pages) are hosted in a web server
- User requests for the initial web page by entering the address on the browser
- Thereafter the user can navigate through the web pages in the site/application using the hyperlinks

# HTML – Structure of HTML document



# HTML – Types of element

- Structural elements
  - header, footer, nav, aside, article
- Text elements
  - Headings – <h1> to <h6>
  - Paragraph – <p>
  - Line break - <br>
- Images
  - 

⋮  
⋮  
⋮  
⋮  
⋮

# HTML – Types of element

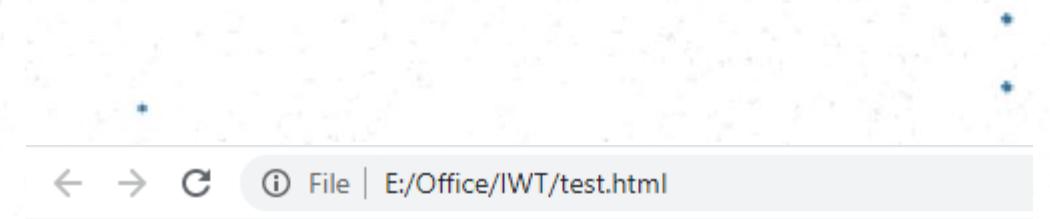
- The HTML elements that doesn't contain any closing tag is referred as "Empty Elements".

`<hr>`  Horizontal Line

`<br>`  Line break

# HTML – First page

```
<!DOCTYPE html>
<html>
 <head>
 <title>My first page</title>
 </head>
 <body>
 <h1>Hello world</h1>
 <p>This is my first html page</p>
 </body>
<html>
```



**Hello world**

This is my first html page

# HTML – Types of element

- Data representational elements (these elements use nested structures)

```
Lists

 IWT
 OOP
 Database

```

Lists

- IWT
- OOP
- Database

```
Lists

 IWT
 OOP
 Database

```

Lists

1. IWT
2. OOP
3. Database

```
tables
<h2>Table</h2>
<table border="1">
 <tr>
 <th>IWT</th>
 <th>OOP</th>
 <th>Database</th>
 </tr>
</table> >
```

## Table

IWT	OOP	Database
-----	-----	----------

# HTML

- You will learn more about these elements and their use in practical class
- Learn more about HTML and HTML5
  - <https://www.w3schools.com/html/default.asp>
  - [https://www.w3schools.com/html/html5\\_intro.asp](https://www.w3schools.com/html/html5_intro.asp)

# Question

- Write html code to display following personal data
  - Name
  - age
  - School

```
<!DOCTYPE html>
<html>
 <head>
 <title>My first page</title>
 </head>
 <body>
 <h1>My name is Saman De Silva</h1>
 <p>I am 70 years old</p>
 <p>My School is ABC college</p>
 </body>
</html>
```

# Summary

- Data networks and the Internet
- Network Services and Protocols
- Web server and the Browser
- Markup languages

IT1100 - Internet and Web Technologies

# Web based Systems development

# Content

-  Distributed systems and their architectures
-  Main concepts of web
-  E-commerce systems and forms

# Distributed systems and their architectures

# Distributed systems and their Architectures

Computer based systems can be mainly divided into 2 types, *according to the distribution of the components*

- Standalone systems (or commonly referred as desktop applications)
- Distributed systems

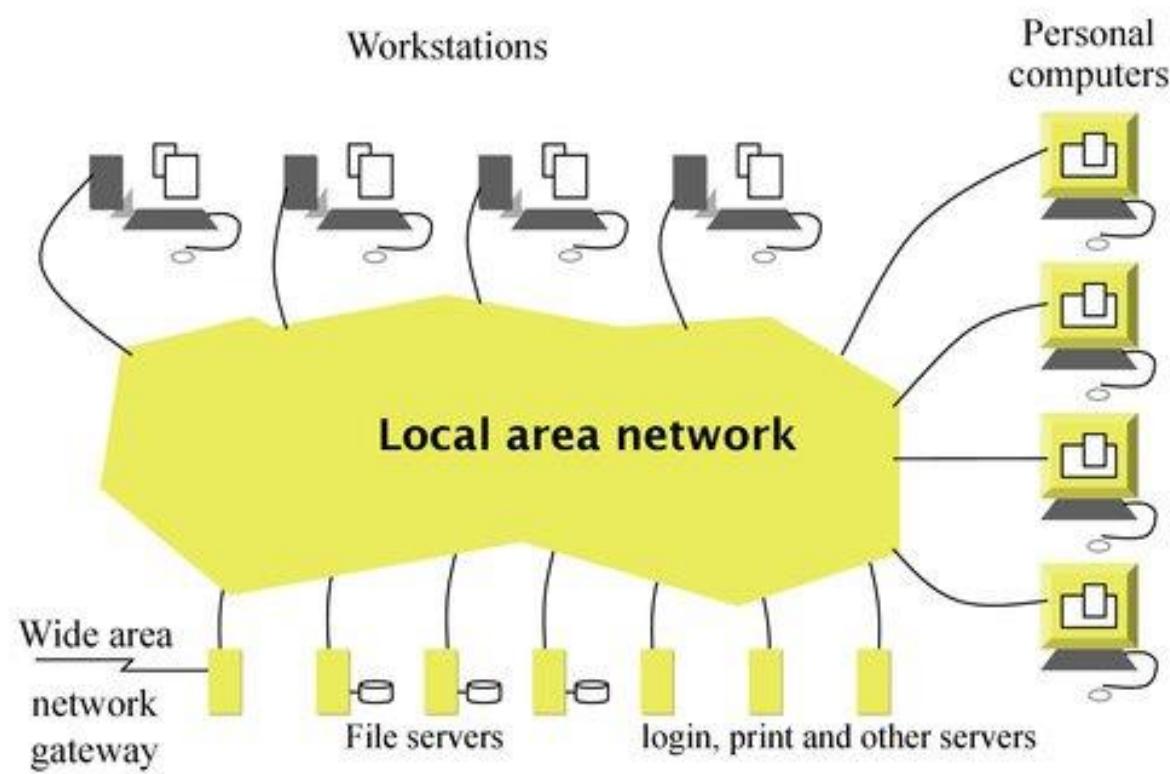
# Distributed Systems vs Standalone Systems

Standalone Computer System	Distributed system
▪ All the components are executed within a single device	▪ The components are distributed and executed in multiple devices
▪ Do not need a network	▪ Need a network
▪ Usually one or tightly coupled set of technologies are used to develop (JAVA, .NET)	▪ Multiple and loosely coupled set of technologies are used to develop (HTML+CSS+JS + PHP) 5

# Distributed systems

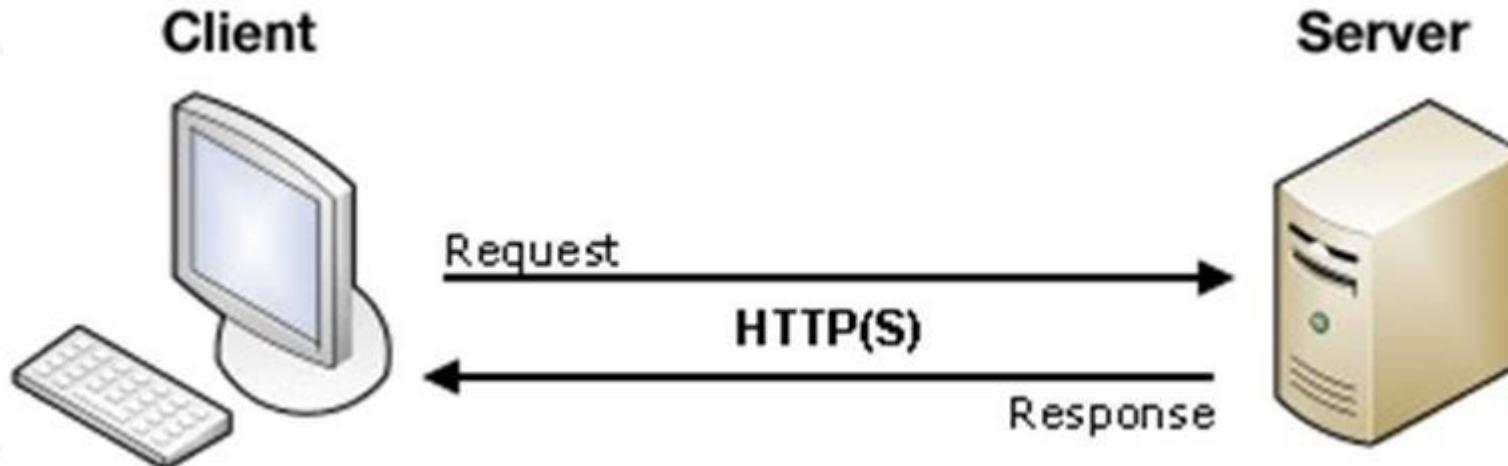
- There are multiple, slightly different definitions and arguments about the terms **distributed systems** and **distributed computing**
- We are here focusing on the systems, whose **components** are **distributed among multiple devices** and **using a network** for the **communication** between these components.

# A Distributed System



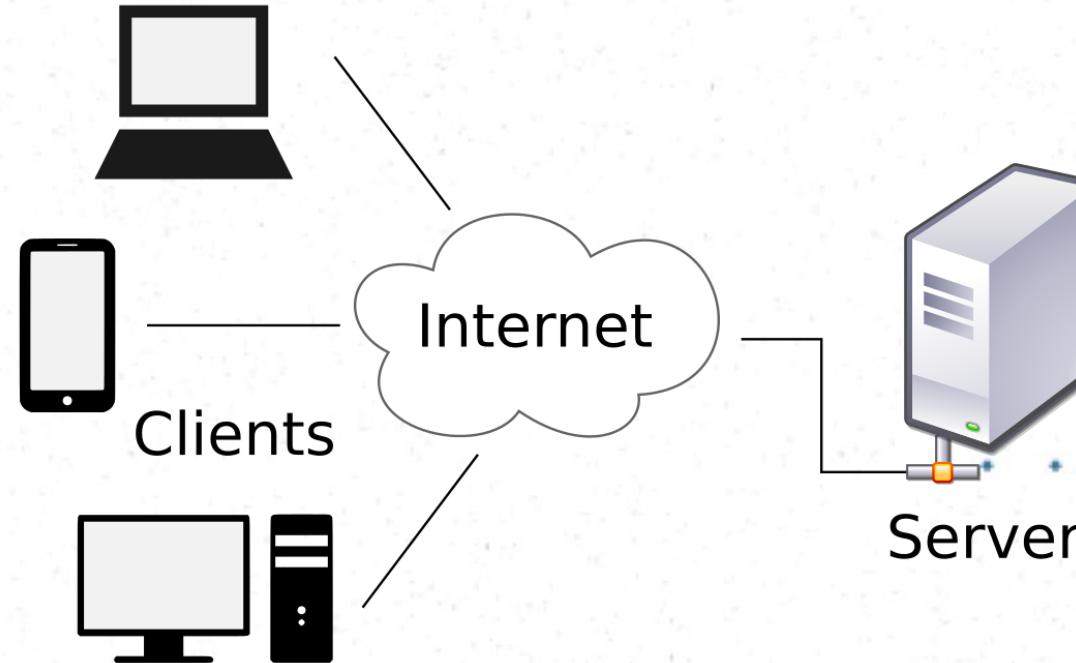
# Client-server architecture (2-tier)

- The basic architecture of the distributed systems is called the client-server (or two-tier) architecture
- Usually the client (user) sends a **request** asking the server for some service and the server **responses** with the resources



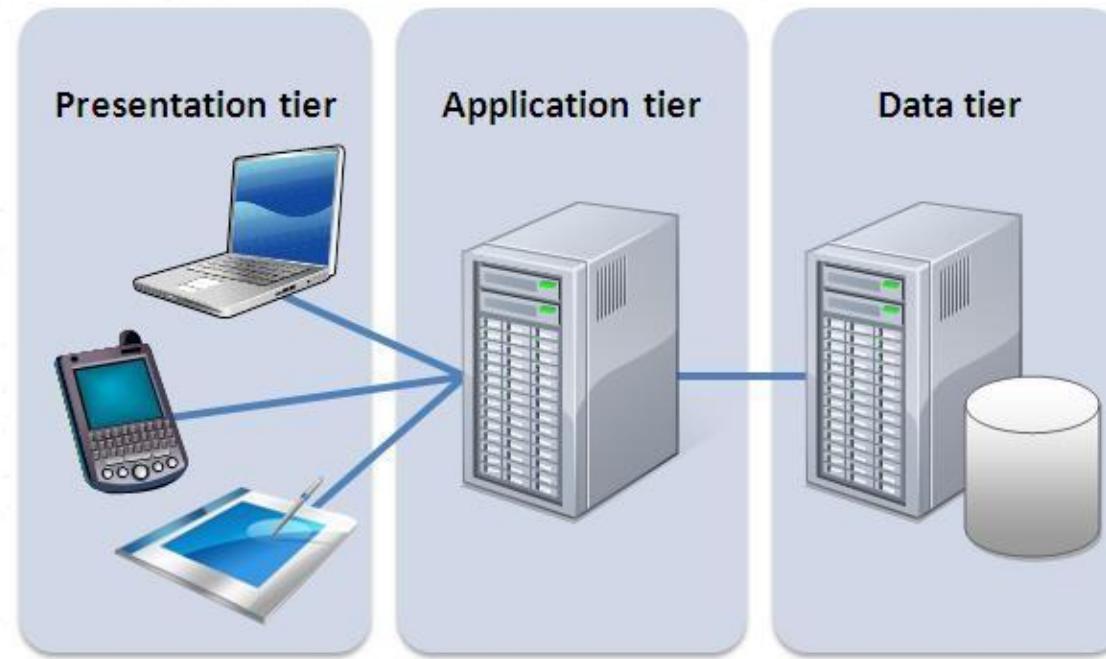
# Client-server architecture (2-tier)

- There can be multiple clients, accessing the same server.
- These clients may use different types of devices



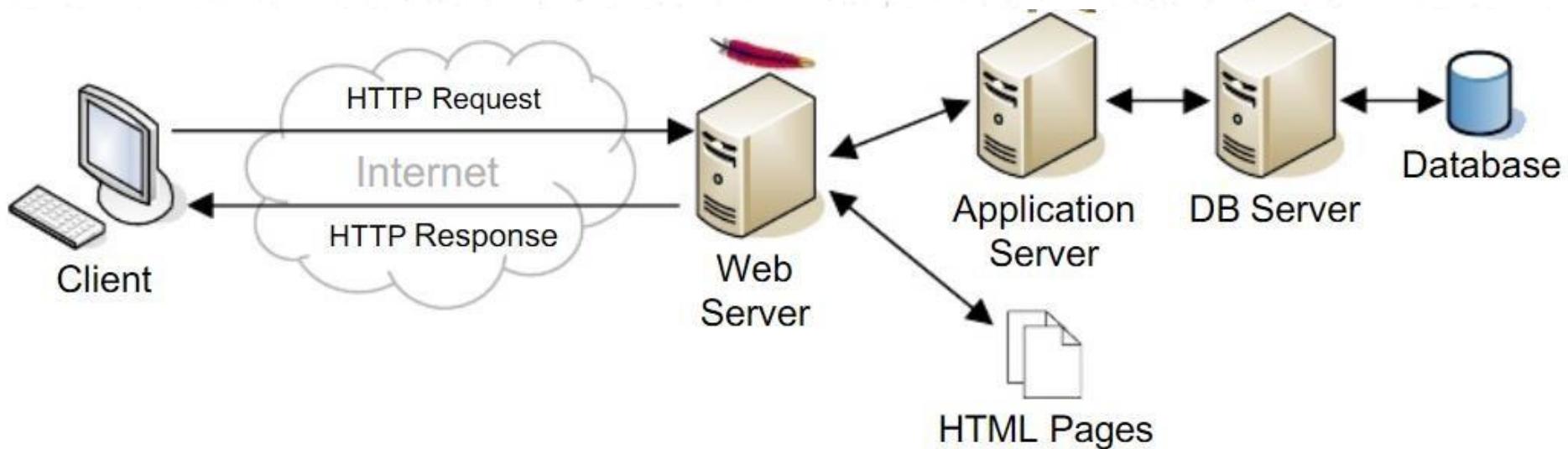
# Client-server architecture (3-tier)

- 3-tier architecture is used, when there is a need for **data persistence** and also to separate the application logic from the data
- This can be seen as an extension of 2-tier architecture



# Client-server architecture (n-tier)

- When there is a need for further separation and distribution of the components, more tiers can be added and extend the 2-tier or 3-tier architecture into an n-tier architecture



# Main concepts of Web

# Web server

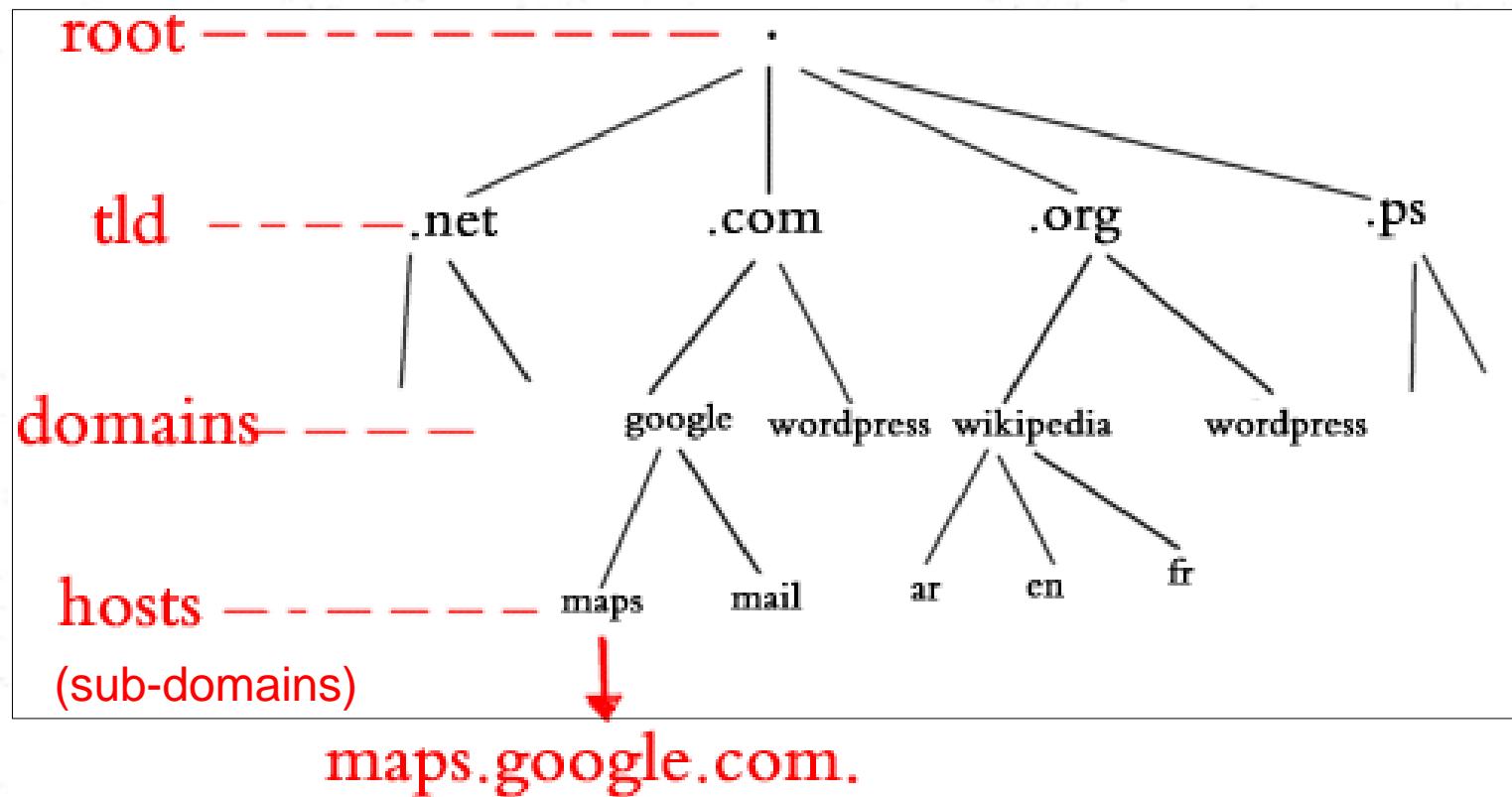
- Web server is a server software, which responds to the HTTP requests.
- Web server means
  - The server software
  - the hardware
  - other software infrastructures
    - which provide a platform to the server software to work and perform well.

# Domain name

- The server computer has an IP address, which is used to access and communicate with the server.
  - Ex: 74.125.236.199
- An IP address is *not human friendly*, therefore more human friendly name is given for humans to identify the server, which is called the **domain name**
  - ex: [www.google.com](http://www.google.com)

# Domain name

There is a hierarchy for the domain names



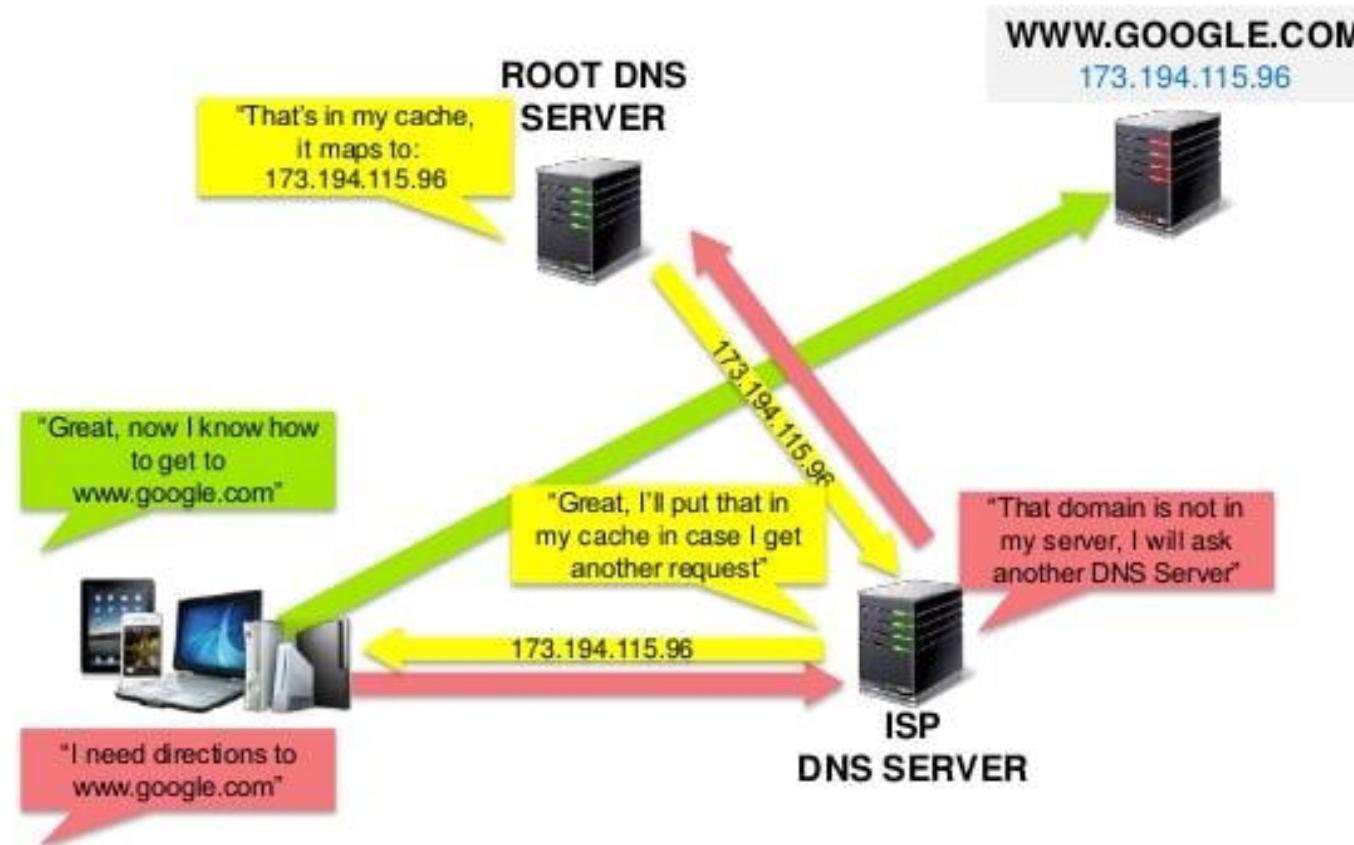
# Domain Name System (DNS)

- DNS is a network, which consists of Domain Name Servers
- DNS helps to map the domain name to the IP address
  - This process is called the address resolution (DNS resolution )

[https://en.wikipedia.org/wiki/Domain\\_Name\\_System](https://en.wikipedia.org/wiki/Domain_Name_System)<sub>17</sub>

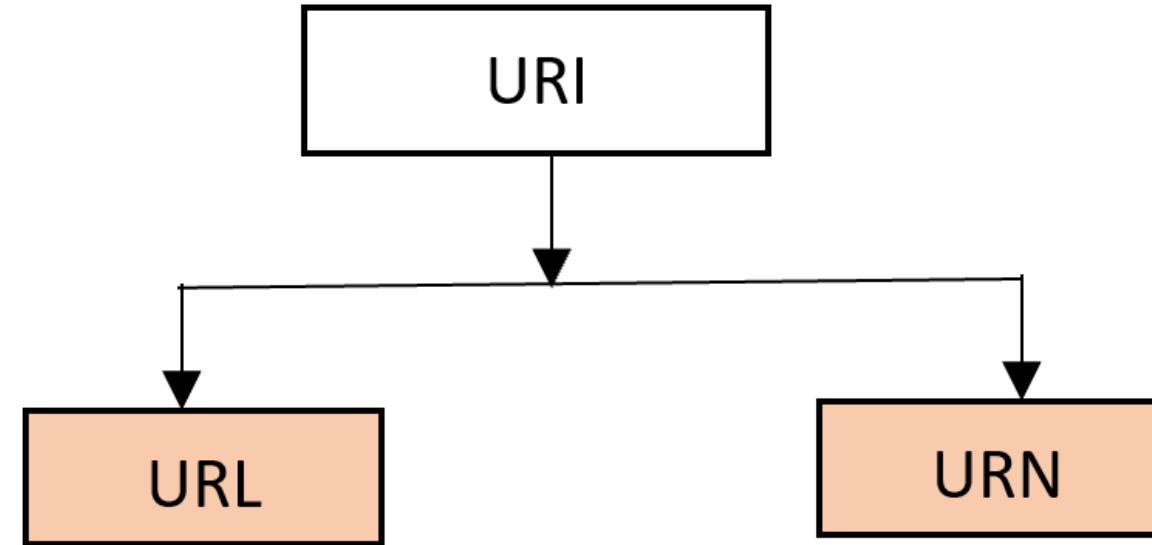
# Domain Name System (DNS)

## How Does DNS Work?



# Unified Resource Identifier (URI)

- URI is a string of characters designed for **unambiguous identification** of resources.
- URI is extensible via the URI scheme
- • • •



## Unified Resource Identifier (URN)

- Unified Resource Name(URN) is a persistent, location-independent identifier

urn:nbn:de:101:3-2019075675872913

urn:uuid:6r4bc420-9c3a-12i9-97d9-0665700c9a66

ISBN 1-446-2776877-40

# Unified Resource Identifier (URL)

Unified Resource Locator (URL) can be seen as a web address, which is a reference to a web resource that specifies its location on a computer network.

- [www.google.com/search?q=examples](http://www.google.com/search?q=examples)
- <https://www.w3schools.com/html/default.asp>
- <https://en.wikipedia.org/wiki/URL>

# URI, URL and URN example

## Difference between URI, URL and URN

http Protocol define  
How to access  
resources

Path of the  
file, directory or  
resource

[http://www.assignmenthelp.net/assignment\\_help/What-is-a-URL](http://www.assignmenthelp.net/assignment_help/What-is-a-URL)

Location where  
resource resides

Resource

**URI:** [http://www.assignmenthelp.net/assignment\\_help/What-is-a-URL](http://www.assignmenthelp.net/assignment_help/What-is-a-URL)

**URL:** [http://www.assignmenthelp.net/assignment\\_help](http://www.assignmenthelp.net/assignment_help)

**URN:** [www.assignmenthelp.net/assignment\\_help/What-is-a-URL](http://www.assignmenthelp.net/assignment_help/What-is-a-URL)

# Websites

- **Website** can be seen as a collection of web pages with **static content**
- Early websites were entirely developed only using HTML
  - Nowadays, some server-side application components and databases are used to dynamically generate the content
  - However, still the content is not user tailored

# Web Applications

- **Web application** is a single page or a collection of web pages, with *interactive components* to *dynamically* generate the content
- Users can enter data., process them, and get information as the result using a web application

**WEBSITE**



**VS**

**WEB APP**



# Websites Vs. Web Applications

- The processing is done by the application components in
  - Client-side (in browser) [JS is used to develop]
  - Server-side (in web server) [PHP, Java can be used]
  - Both the client and the server sides
- These application components may allow the user to interact with the web application by entering data, selecting content, clicking, dragging and dropping, etc...

# E-commerce systems and forms

# E-commerce systems and forms

Nowadays, the web technologies are used to develop many types of applications, including

- Email (Gmail, yahoo mail, outlook)
- Office tools (Google docs, google sheets, etc..)
- Games (Poki, Miniclip)
- Multimedia (YouTube)
- Social media and networking (Facebook, Tweeter)
- E-commerce (Amazon, e-bay)
- And many more.....

# E-commerce systems

E-commerce is a large domain, which covers many related concepts like

- Internet marketing
- Electronic fund transfer
- Online transaction processing
- And many more.....

# E-commerce systems

E-commerce systems provide online buying and selling over the internet.

There is a large variety of types of e-commerce systems

- Online goods/soft items(software, e-books, videos)
- Retail services (travel, food, cloths)
- Marketing services (advertising, auctions)
- Customer services (help centers, online banking)
- Many more....

# E-commerce systems

- E-commerce systems can also be categorized according to the stakeholder engagement

- Business-to-Business (B2B) – between companies
- Business-to-Consumer (B2C) – traditional operations
- Consumer-to-Consumer (C2C) – via an online platform

# Advantages of e-commerce

- To customers
    - No queues
    - Reduced price
    - Global transactions
    - Available 24/7
    - Wide collection for easy selection

# Disadvantages of e-commerce

- To customers
  - Cannot examine the product
  - Lacks the shopping experience
  - Needs internet access
  - Can be addicted

# Advantages of e-commerce

- *To businesses*
    - After the capital cost, maintenance cost is low
    - Global customers
    - Increased market share

# Disadvantages of e-commerce

- *To businesses*

- For physical items, storing and distributing is needed
- Need to update the system frequently
- Depends on the power and the internet

⋮ ⋮ ⋮

# Web Forms

- E-commerce systems use web forms to capture the customers' data and send to the server for processing

The screenshot shows a "Checkout" page with a "Shipping Information" section and an order summary.

**Shipping Information:**

- First Name: [Input field]
- Last Name: [Input field]
- Address Line 1: [Input field]
- Address Line 2: [Input field]
- Address Line 3: [Input field]
- City: [Input field]
- Region: [Input field]
- Postal Code: [Input field]
- Country: United Kingdom (dropdown menu)
- Primary Phone Number: [Input field] (highlighted with a red border)
- Alternate Phone Number: [Input field]
- Email Address: [Input field]

**Order Summary:**

Items:	GBP 60.50
Shipping:	GBP 30.99 GBP 25.00
Duties & Taxes:	GBP 11.66 INCLUDED
Order Total:	GBP 85.50

At the top left, there is a message: "✓ You received 25 GBP flat-rate shipping!"

# Web Forms

- Forms use variety of fields (elements) or structures not only to capture the users' data, but also to display data and information
- Input fields
  - Text boxes, dropdown lists, option buttons, radio buttons, selectable items, drag and drop items, file selectors, etc..
- Data/information display structures
  - Lists, tables, charts, images, files, etc..

# Web Forms Development

- When developing HTML web forms, all the form elements are wrapped by the parent element named “form”, which has 2 main attributes

```
: : : : : <form method="get" action="controller.php">
: : : : : {
: : : : : | The form is developed
: : : : : | inside the form element
: : : : : }
: : : : : </form>
```

## *Form method*

- Used to specify the type of the intended action the form submission is needed

Ex: available form methods in HTML

- When submitting data, the form method specifies the way the data should be submitted
  1. GET
  2. POST

## *Form method – get*

- Default form method to submit data
- Data is visible on the address bar
  - [www.myDomain.com/controller.php?name=Saman&age=35](http://www.myDomain.com/controller.php?name=Saman&age=35)
- Appends form-data into the URL in name/value pairs (Query string)
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
  - – GET is better for non-secure data
  - Useful for form submissions where a user want to bookmark the result

## *Form method – post*

- Use POST method to send sensitive or personal information.
  - The POST method does not display the submitted form data in the page address field.
- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

## *Form action*

- Specifies the address (URL) to submit the form
  - Usually, a script file or a program
- This address points to a component in the server
- When the form is filled and submitted, this component will receive the form values then process and responds with the necessary output

# Form submission

- There should be a submit button to submit the form to the action end, using the specified form method

```
<input type="submit" value="Submit">
```

# *Form elements*

- Many form fields are developed using the input element

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

- You will learn more possible form elements and their use in the practical session

[https://www.w3schools.com/html/html\\_forms.asp](https://www.w3schools.com/html/html_forms.asp)

```
<fieldset>
<legend>Personal information:</legend>

<input type="button" value="Button 1">

<input type="checkbox">value1

<input type="color">

<input type="date">

<input type="datetime-local">

<input type="email">

<input type="file">

<input type="hidden">

<input type="image">

<input type="month">

<input type="number">

<input type="password">

<input type="radio">option 1

<input type="range">

<input type="reset">

<input type="search">

<input type="submit">

<input type="tel">

<input type="text">

<input type="time">

<input type="url">

<input type="week">

</legend>
</fieldset>
```

Personal information:

The screenshot shows a web page with a form titled "Personal information:". The form includes the following elements:

- A button labeled "Button 1".
- A checkbox labeled "value1".
- A color picker.
- A date input field.
- A datetime-local input field.
- An email input field.
- A file input field with the placeholder "Choose File" and the value "No file chosen".
- A submit button labeled "Submit".
- A text input field with the placeholder "-----, -----".
- A radio button labeled "option 1".
- A range input field.
- A reset button labeled "Reset".
- A submit button labeled "Submit".
- A time input field.
- A url input field.
- A week input field.

# *Form validation*

- It is very important to validate the data values entered into a form, before processing them
- Form validation can be done
  - Using HTML5 in **client-side** (before submitting the form)
  - Using JS in **client-side** (before submitting the form)
  - Using a **server-side** component (usually the component pointed by the form action) in the server (After submitting the form)

# *Form validation*

## Form validation – using HTML

```
<input type="text" name="name" required>
<input type="text" name="name" value="Sam" readonly>
<input type="text" name="name" value="Sam" disabled >
.....
<input type="text" name="name" value="Sam" size="20" >
<input type="text" name="name" value="Sam" maxlength="50" >
.....
```

- You will learn the use of JS and PHP to validate form data, when you learn JS and PHP

# Summary

- Distributed systems and their architectures
- Main concepts of web
- E-commerce systems and forms

IT1100 - Internet and Web Technologies

## Lecture 03

# Basics of the client-side development

# Content

- What is CSS and Why CSS?
- CSS statements with Properties and Values
- How to use CSS?
- Advanced selectors

# What is CSS and Why CSS?

# Introduction

- Development of the **browser-based** web application
  - uses 3 main technologies in the client-side
    - 1. HTML
      - To develop the content
    - 2. CSS
      - To format/decorate the content
    - 3. JavaScript (JS)
      - To develop the application (processing) components

# CSS

- CSS – Cascading Style Sheets

- Used to

- Decorate / Format web page content

# CSS and HTML

HTML



HTML+CSS



# Advantages of using CSS

- Reduce HTML formatting tags
- Easy modification
- Save lot of work and time
- Faster loading

⋮  
⋮  
⋮  
⋮

⋮  
⋮  
⋮  
⋮

# Without CSS

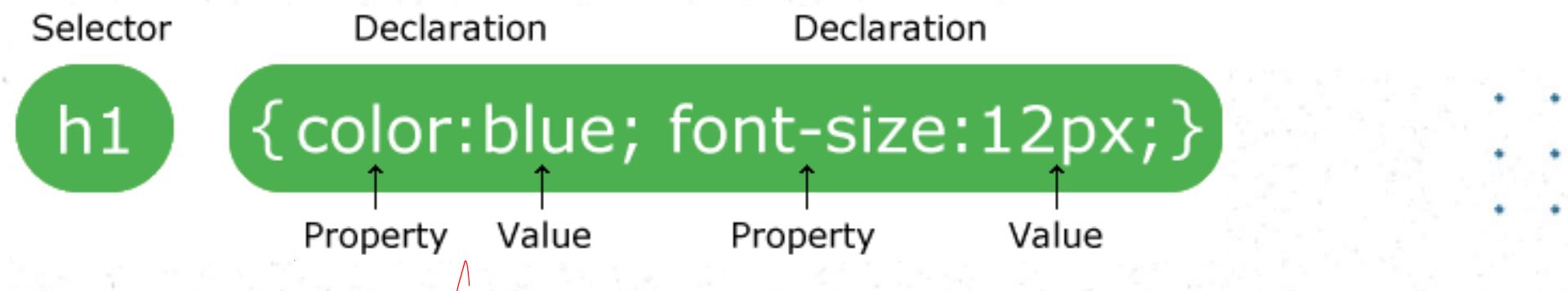


# With CSS



# CSS statements

# Selector, Properties and Values



- 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.

# Selector, Properties and Values

- Examples

```
body {
 background-color: yellow;
}

h1 {
 color: purple;
 text-align: center;
}

p {
 font-family: tahoma;
 font-size: 25px;
}
```

My First CSS Example  
This is a paragraph.

```
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
```

# Some formatting categories

- Positioning
- Size
- Alignment
- Font / Text
- Color / Background / Border

# Activity

Find the available CSS rules under the following categories

- Positioning
- Size
- Alignment
- Font / Text
  - [https://www.w3schools.com/css/css3\\_fonts.asp](https://www.w3schools.com/css/css3_fonts.asp)
- Color / Background / Border
  - [https://www.w3schools.com/css/css3\\_backgrounds.asp](https://www.w3schools.com/css/css3_backgrounds.asp)

# How to use CSS?

1. Inline
2. Internal sheets
3. External sheets



# How to use CSS?

# Inline

- defined within the "**style**" attribute of the relevant element

⋮  
⋮  
⋮

- Used as HTML attribute

```
<tag style = “CSS Statements” >
```

⋮  
⋮  
⋮

# Inline

## *Examples*

- <h1 style="color:red;">Heading 01</h1>
- <p style="color:blue; font-family: Tahoma;">  
IWT Lecture 03  
</p>

# Question 1

```
<h1>my timetable</h1>
```

Ex. Apply below CSS rules to the element above

- text-align: center;
- color: #0000CC;
- font-family: Tahoma, Geneva, sans-serif;
- font-size: 32px;

```
<!DOCTYPE html>

<html>

<head>

</head>

<body>

• •
• <h1 style="text-align:center; color:#0000CC;
font-family:Tahoma, Geneva, sans-serif; font-
size:32px;">my timetable </h1>
• •
• </body>

</html>
•
```



Output

**my timetable**

•  
•  
•  
•

# Inline

- <h1 style="text-align:center; color:#0000CC; font-family:Tahoma, Geneva, sans-serif; font-size:32px;">**my timetable**</h1>
- <h1 style="text-align: center; color:#0000CC; font-family: Tahoma, Geneva, sans-serif; font-size: 32px;">**my timetable**</h1>

This is more  
readable

# Internal style sheet

- Why not inline CSS?
  - Difficult to modify

⋮ ⋮  
⋮ ⋮  
⋮ ⋮

- Solution
  - Internal style sheet

⋮ ⋮ ⋮

# Internal style sheet

- Where to write?

1. In the **head** section of the HTML

document

2. Using the **style** element

```
<style></style>
```

# Internal style sheet

- How to write?

```
<head>
 <style>
 Selector
 {
 /* CSS Statements */
 }
 </style>
</head>
```

# Internal style sheet

- We use **selectors** to access the content
- Selectors
  - 1. By Element (h1, p, a)
  - 2. By ID (#MainHeading, #Section1)
  - 3. By Class (.Menu, .Header, .Footer)

# Internal style sheet [By element]

```
<!DOCTYPE html>
<html>
<head>
<style>
 a {
 color:#F00;
 }
</style>
</head>
<body>

 visit SLIIT

</body>
</html>
```



Output

visit SLIIT

# Internal style sheet [By ID]

- CSS in **head** <style></style>

```
#ParaMain{
 color:#F00;
}
..
```

- Use the **ID** attribute for the element in the **body**

- ID should be unique

```
<p id="ParaMain">My paragraph </p>
```

# Internal style sheet [By ID]

```
<!DOCTYPE html>
<html>
<head>
<style>
#ParaMain{
 color:#F00;
}
</style>
</head>
<body>

<p id="ParaMain">My paragraph </p>
<p>My paragraph </p>
<p>My paragraph </p>

</body>
</html>
```

Output

My paragraph

My paragraph

My paragraph

# Internal style sheet [By class]

- CSS in **head** <style></style>

```
.MainFont
{
 font:sans-serif;
 font-size:12px;
}
```

- Use the **class** attribute for the element(s) in the **body**

- There can be multiple element, who belong to the same class

```
<p class="MainFont">My paragraph </p>
```

```
Google
```

# Internal style sheet [By class]

```
<!DOCTYPE html> <html> <head>
 <style>
 .MainFont_1
 {
 font-size:12px; color:blue;
 }
 .MainFont_2
 { font-size:12px; color:red; }
 </style>
 </head> <body>
 <p class="MainFont_1">My paragraph </p>
 Google
 Google
 </body> </html>
```



Output

My paragraph

Google Google

# External style sheets

## Why External?

- Easy to modify
- Uniform formatting in the site
- Design themes
- Faster loading

•  
•  
•  
•

# External style sheets

CSS is coded in an **external** file

- Extension of the file should be **.css**
- The code in the file starts with the line

```
@charset "utf-8";
```

### 3.3. External style sheets

```
@charset "utf-8";
/* CSS Comments*/

...
.MainFont
{
 font: Tahoma, Geneva, sans-serif;
 font-size: 12px;
}
```

# External style sheets

How to link the external CSS file with the web page?

- within the **<link>** element, inside the **<head>** section

```
<head>
 <link rel="stylesheet" href="CSSMain.css">
</head>
```

```
<!DOCTYPE html>
<html>
<head>
 <link rel="stylesheet" href="html_external.css">
</head>
<body>
 <p class="MainFont">My paragraph </p>
 Google
 Google
</body>
</html>
```

```
.MainFont
{
 font: Tahoma, Geneva, sans-serif;
 font-size: 12px;
}
.MainFont_1
{
 font:Tahoma, Geneva, sans-serif;
 font-size:12px;
 color:blue;
}
.MainFont_2
{
 font:Tahoma, Geneva, sans-serif;
 font-size:12px;
 color:red;
}
```

### Output

My paragraph  
[Google](http://www.google.com) [Google](http://www.google.com)

# Advanced Selectors

# Pseudo Elements

- A CSS pseudo-element is used to style specified parts of an element.
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element

# Pseudo Elements

## Notation

```
Selector::pseudo-element {
 /* CSS Statements */
}
```

```
p::first-line {
 color: red;
}
```

Year one Semester 02 modules.  
There are 4 modules.

<p>Year one Semester 02 modules. <br>There are 4 modules. </p>



# Pseudo Classes

- A CSS pseudo-element is used to define a special state of an element.
- For example, it can be used to:
  - Style an element when a user mouses over it
  - Style visited and unvisited links differently

# Pseudo Classes

## Notation

```
Selector:pseudo-class {
 /* CSS Statements */
}

/* mouse over link */
a:hover
{
 background-color: red;
}
```

# Example 01

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

</style>
</head>
<body>

 <h1>This is a heading</h1>
 <h2>This is a smaller heading</h2>
 <p>This is a paragraph.</p>
 <p>This is another paragraph.</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, p {
 color: red;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<h2>This is a smaller heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

This is a heading

This is a smaller heading

This is a paragraph.

This is another paragraph.

# Example 02

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

</style>
</head>
<body>

<h1>This is a Heading</h1>
<p id="para1">This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
 color: red;
}
</style>
</head>
<body>

<h1>This is a Heading</h1>
<p id="para1">This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

**This is a Heading**

This is a paragraph.

This is another paragraph.

# Exercise

```
<!DOCTYPE html>
<html>
<head> <style>
h1 {
 text-transform: uppercase;
}
p {
 text-transform: capitalize;
}
</style> </head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

Result:

**THIS IS A HEADING**

This Is A Paragraph.

This Is Another Paragraph.

# CSS – Summary

- What is and Why CSS?
- CSS statements with Properties and Values
- How to use CSS?
- Advanced selectors

⋮  
⋮  
⋮  
⋮

⋮  
⋮  
⋮  
⋮

IT1100 - Internet and Web Technologies

# Lecture 04

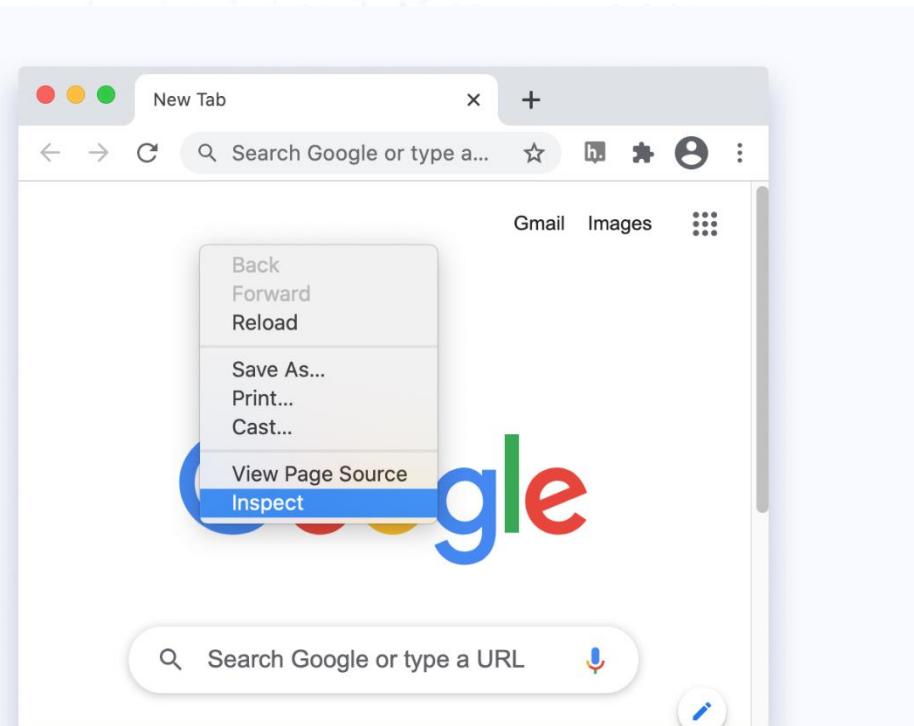
## JavaScript – Part 1

# Content

- Introduction to the JavaScript
  - Variables in JS
  - Operators in JS
  - Control structures in JS
- .....

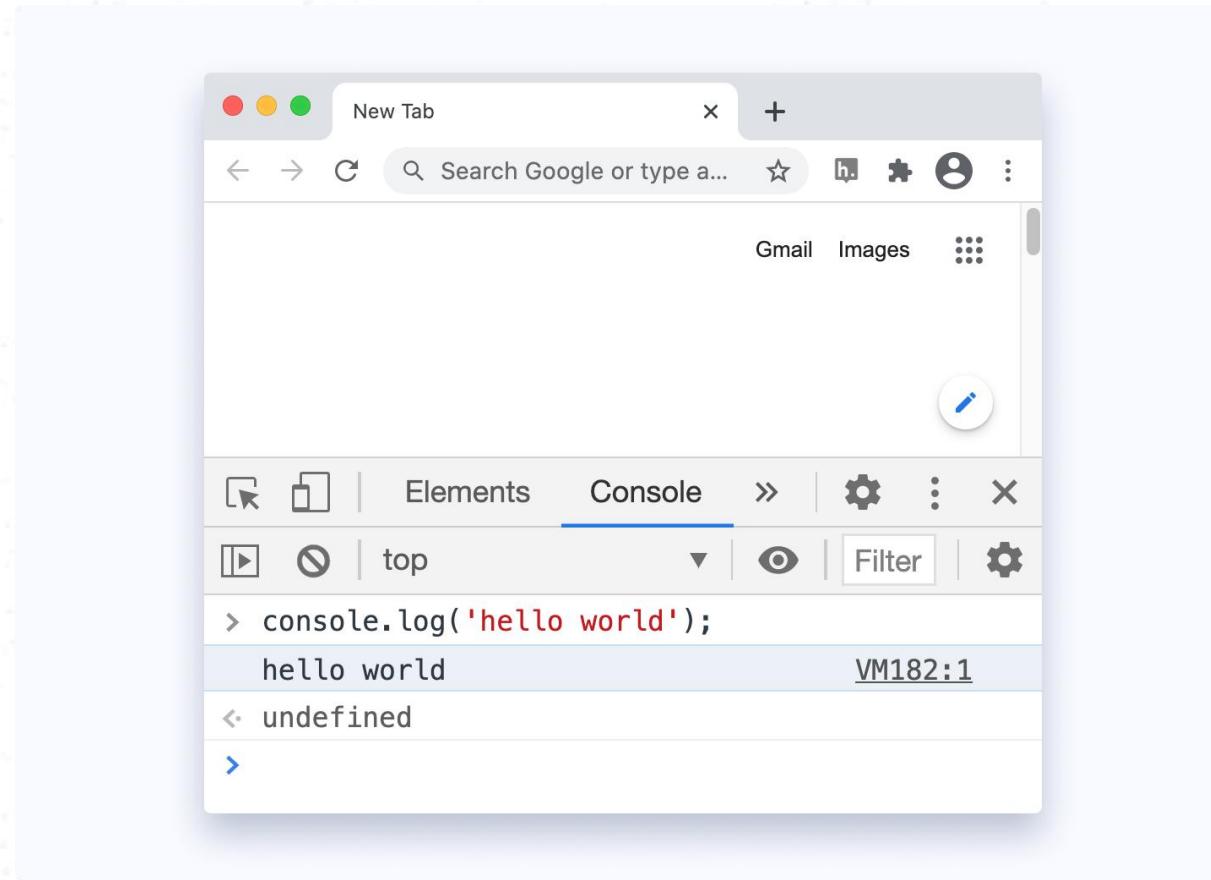
# Using Console Tab of Web Browsers

- All the popular web browsers have built-in JavaScript engines. Hence, you can run JavaScript on a browser. To run JavaScript on a browser,
- Open your favorite browser (here we use Google Chrome).
- Open the developer tools by right clicking on an empty area and select Inspect. Shortcut: F12.



# Using Console Tab of Web Browsers

- On the developer tools, go to the console tab. Then, write JavaScript code and press enter to run the code.



# Why JavaScript?

HTML



JS



CSS



# Methods of using JavaScript

# Methods of using JavaScript

## 1. Internal Script

- Scripts in the <body> section
- Scripts in the <head> section

## 1. External Script files

# Internal script

- JavaScript is embedded into the HTML document using the **script** element

```
<script >
```

```
//JS code
```

```
</script>
```

⋮  
⋮  
⋮  
⋮

# Internal script Example in the <head> section

```
<html>
<head>
<title>Internal JavaScript</title>
<script>
 document.write("Internal JavaScript in the head section");
</script>
</head>

<body>
 <h3 style="color:red;"> JavaScript </h3>
</body>
</html>
```

output

Internal Javascript in the head section

**JavaScript**

# Internal script Example in the <body> section

```
<html>
<head>
<title>Internal JavaScript</title>
</head>
<body>
<h3 style="color:purple;"> JavaScript </h3>
<script>
document.write("Internal JavaScript in the body section");
</script>
</body>
</html>
```

output

JavaScript

Internal JavaScript in the body section

# External script

- The external JS file should use the extension as .js
- External file is linked to the web page in head using the **script** element
- The script element uses the **src** attribute to specify the URL of the source JS file
  - src=“<Location>/<FileName>.js”***

# External script

## External script file linking

```
<head>
 <script src=".../ClientScripts/MainJS.js"></script>
</head>
```

- The same JS file can be linked to multiple pages

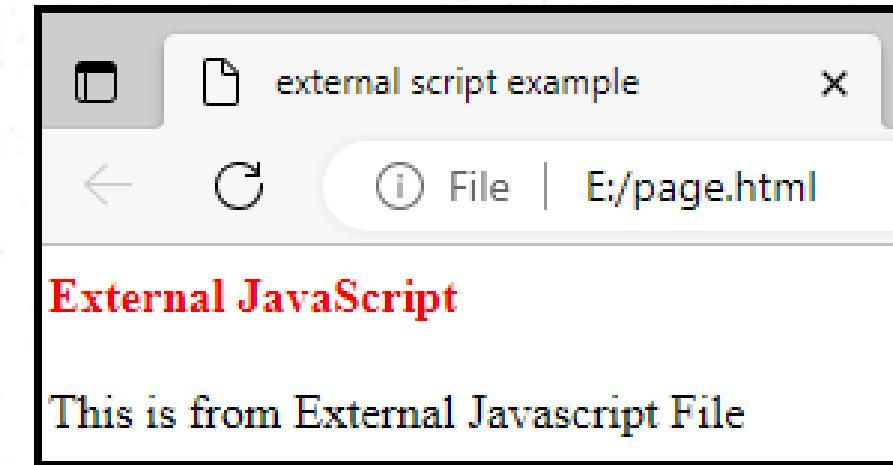
# External JavaScript Example

```
<html>
<head>
 <title>
 external script example
 </title>
</head>
<body>
 <h4 style="color:red">External JavaScript</h4>
 <script src="ex1.js"></script>
</body>
</html>
```

Page.html

document.write("This is from External Javascript File");

output



ex1.js

# Variables in JS

# Data types in JS

## 1. Numerical

- Integers – 1, 2, 3, -56, -135, 3464
- Floating point/Decimal – 34.46, -65.135

⋮ ⋮  
⋮ ⋮  
⋮ ⋮

## 2. Strings

- Single characters – “a”, “b”, “c”, “2”, “7”
- Multiple characters – “abc”, “12/04/2012”, “34”

⋮ ⋮  
⋮ ⋮  
⋮ ⋮

## 3. Boolean – true / false

## 4. Null

## 5. Undefined

⋮ ⋮ ⋮ ⋮

# Data types

## Note:

- JavaScript does not make a distinction between integer values and floating point values.
- All numbers in JavaScript are represented as floating-point values.

# Variable declaration

The keyword **var** and **let** is used to declare a variable

- Examples

- var age;
- var smallNumber;
- var initial
- var name
- var isPassed;
- var num1, num2, num3;



# Standars for the variable name

- You should not use any of the JavaScript **reserved keywords** as a variable name.

- No spaces
- Meaningful
- Use camel case

abstract	else	Instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

# Variable Initialization

```
var age = 20;
var height = 5.5;
var initial = "K";
var name = "Kamal";
var isPassed = true;
```

⋮  
⋮  
⋮  
⋮

# Assign values to the variables

age = 20;

height = 5.5;

initial = "K";

name = "Kamal";

isPassed = true;

# JavaScript Constant Variables

The **const** keyword was also introduced in the **ES6(ES2015)** version to create constants. For example,

```
const x = 5;
x = 10; // Error! constant cannot be changed
console.log(x)
```

Also, you cannot declare a constant without initializing it. For example

```
const x; // Error! Missing initializer in const declaration
x = 5;
console.log(x)
```

# Read and use the variable value

```
var age = 20; //Declare and initialize the variable
document.write(age);

age = 25; //Assign a new value to the variable
document.write("
Modified age = " + age);
```



JS is a  
weakly  
typed  
language

```
<html>
<head><title>JavaScript Page</title> </head>
<body><script type="text/javascript">
 document.write("4"/3); → 1.33333333333333
 document.write("
");
 document.write("5" +5); → 55
 document.write("
");
 document.write("5"- 3); → 2
 document.write("
");
 document.write("5"*"5"); → 25
 document.write("
");
 document.write(4*3); → 12
 document.write("
");
 document.write(5* "5"); → 25
</script>
<h1>Hello world</h1>
</body>
</html>
```

output

Hello world

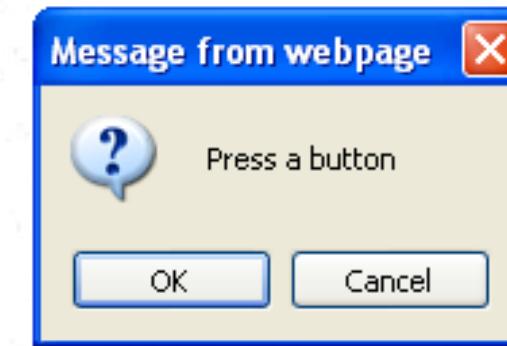


# JavaScript popup boxes

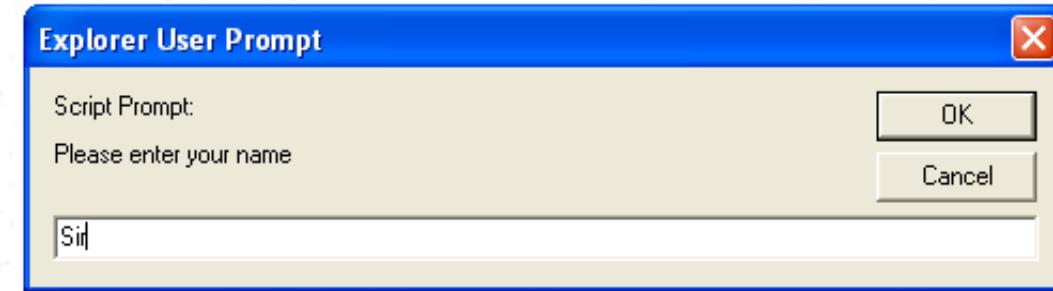
- Alert box
  - `alert ("This is an important message !");`



- Confirm box
  - `var response=confirm("Press a button");`



- Prompt box
  - `var name=prompt("enter your name","Sir");`



# Operators in JS

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators

# Arithmetic Operators

Operator	Description	Example	Result
+	Addition	$x=y+2$	$x=7$
-	Subtraction	$x=y-2$	$x=3$
*	Multiplication	$x=y*2$	$x=10$
/	Division	$x=y/2$	$x=2.5$
%	Modulus (division remainder)	$x=y\%2$	$x=1$
++	Increment	$x=++y$	$x=6$
--	Decrement	$x=--y$	$x=4$

# Assignment Operators

Operator	Example	Same As	Result
=	$x=y$		$x=5$
$+=$	$x+=y$	$x=x+y$	$x=15$
$-=$	$x-=y$	$x=x-y$	$x=5$
$*=$	$x*=y$	$x=x*y$	$x=50$
$/=$	$x/=y$	$x=x/y$	$x=2$
$%=$	$x\%=y$	$x=x \% y$	$x=0$

# Comparison Operators

Operator	Description	Example
<code>==</code>	is equal to	<code>x==8</code> is false
<code>====</code>	is exactly equal to (value and type)	<code>x====5</code> is true <code>x===="5"</code> is false
<code>!=</code>	is not equal	<code>x!=8</code> is true
<code>&gt;</code>	is greater than	<code>x&gt;8</code> is false
<code>&lt;</code>	is less than	<code>x&lt;8</code> is true
<code>&gt;=</code>	is greater than or equal to	<code>x&gt;=8</code> is false
<code>&lt;=</code>	is less than or equal to	<code>x&lt;=8</code> is true

# Logical Operators

Operator	Description	Example
<code>&amp;&amp;</code>	and	<code>(x &lt; 10 &amp;&amp; y &gt; 1)</code> is true
<code>  </code>	or	<code>(x==5    y==5)</code> is false
<code>!</code>	not	<code>!(x==y)</code> is true

# Control Structures in JS

# **Selection / Branching**

- Simple if-else
- If-else ladder
- Nested if-else
- Switch



# **Repetition / Iteration / Looping**

- While loop
- For loop



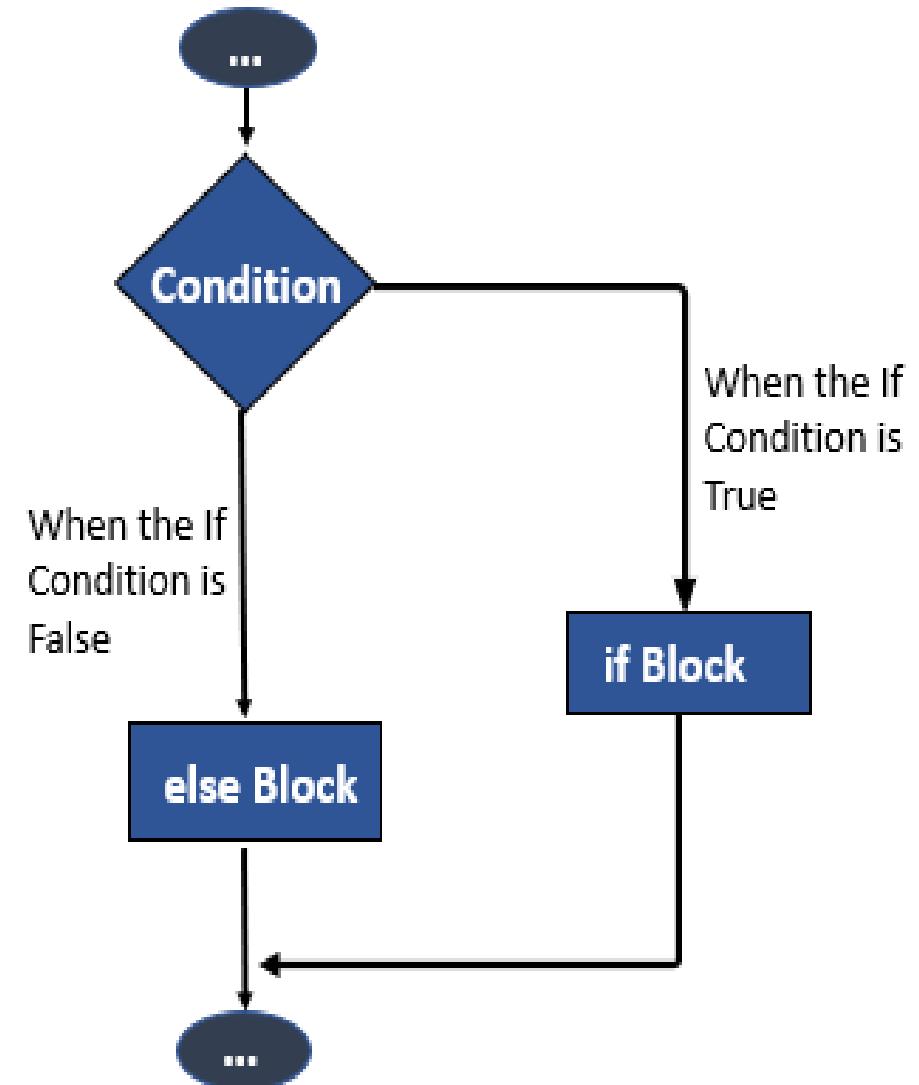
# Simple if-else

- Used to divide the algorithm execution path into branches based on conditions
- Conditions produce Boolean results
- If the condition is true – we can do something
- Else we can do some other thing

# Simple if-else

```
if (<Condition>)
{
 //Do something
}
else
{
 //Do some other thing
}
```

- Else is optional



# Simple if-else example

- User enters the mark for Maths.
    - If the mark is greater than or equals 50 then display a message “Pass”
    - Else display a message “Fail”

# Simple if-else example

```
if(mark >= 50)
{
 document.write ("Pass");
}
else
{
 document.write ("Fail");
}
```

# Simple if-else example

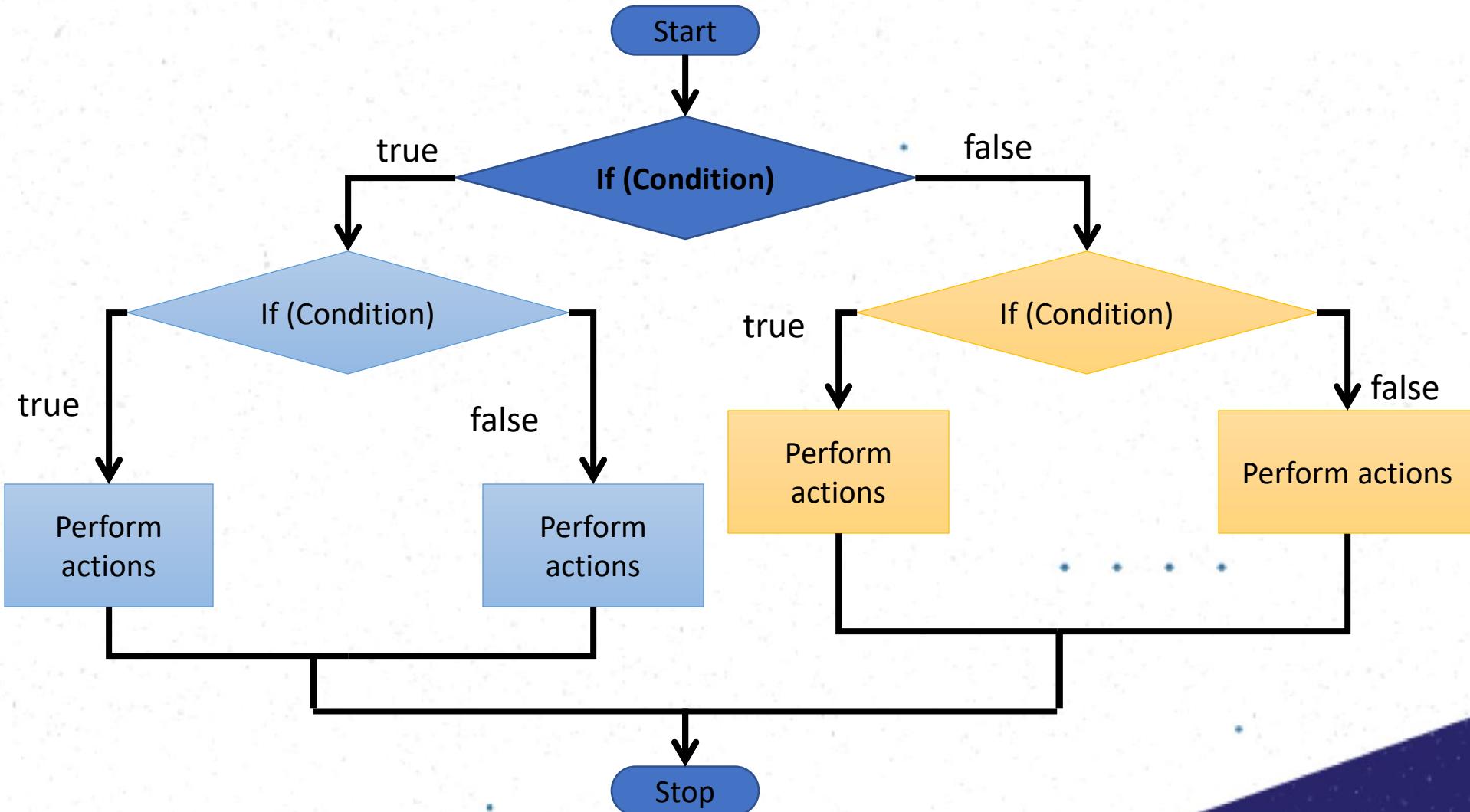
```
// check is the number is positive or negative/zero
const number = prompt("Enter a number: ");

// check if number is greater than 0
if (number > 0) {
 console.log("The number is positive");
} // if number is not greater than 0

else {
 console.log("The number is either a negative number or 0");
}

console.log("The if...else statement is easy");
```

# Nested if-else



# Nested if-else

```
if(<Condition1>
{
 if(<Condition2>) { //Actions }
 else { //Actions }
}
else
{
 if(<Condition3>) { //Actions }
 else { //Actions }
}
```

⋮  
⋮  
⋮

⋮  
⋮  
⋮

# Nested if-else

- Are these equivalent?

```
if (age < 12) {
 entry = "free";
} else if (age < 18) {
 entry = "£10";
} else {
 entry = "£20";
}
```

```
if (age < 18) {
 entry = "£10";
} else if (age < 12) {
 entry = "free";
} else {
 entry = "£20";
}
```

• •  
• •  
• •

# Nested if-else example

## output

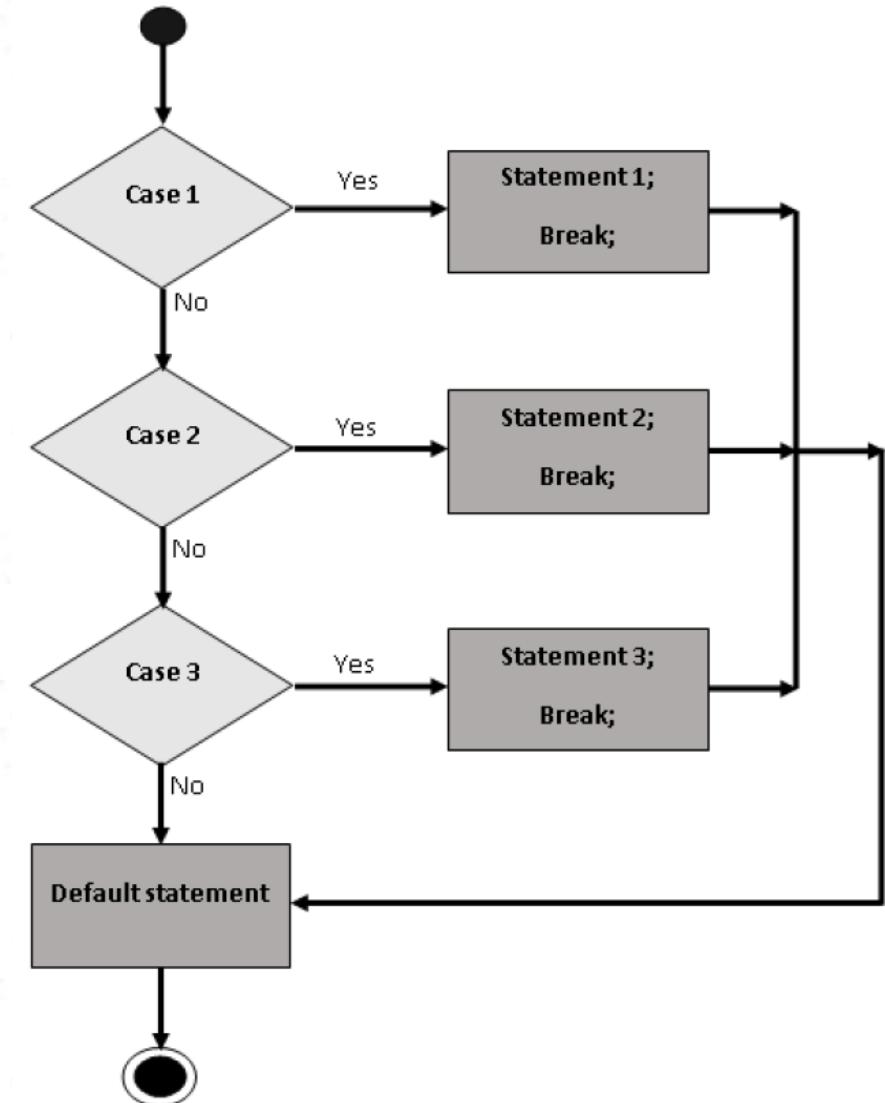
Good day

This example demonstrates the if..else if...else statement.

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time<10)
{
 document.write("Good morning");
}
else if (time>=10 && time<16)
{
 document.write("Good day");
}
else
{
 document.write("Hello World!");
}
</script>
<p>
This example demonstrates the if..else if...else statement.
</p>
</body>
</html>
```

# Switch

```
switch(n)
{
 case 1:
 execute code block 1
 break;
 case 2:
 execute code block 2
 break;
 default:
 code to be executed if n is different
 from case 1 and 2
}
```



# Switch example

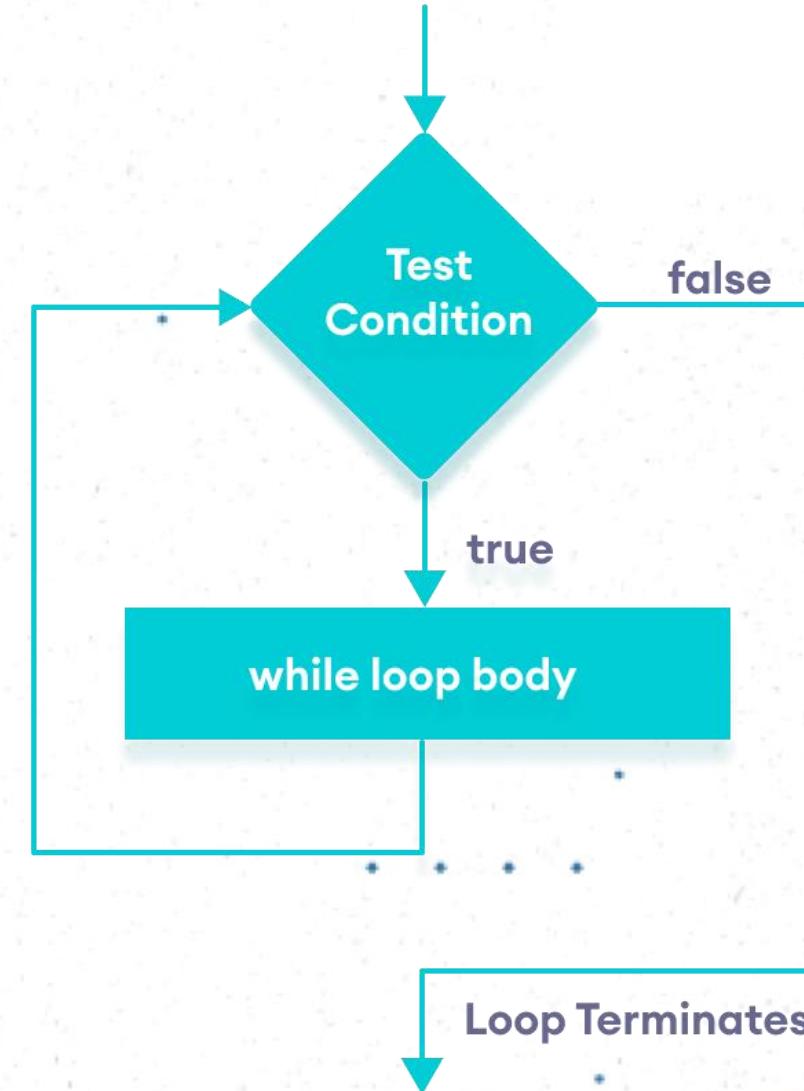
```
var grade="B";
switch (grade)
{
 case "A":
 alert("Excellent");
 break;
 . . .
 . . .
 case "B":
 alert("Good");
 break;
 . . .
 . . .
 default:
 alert("Average");
 break;
}
```

# Switch example

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
 document.write("Finally Friday");
 break;
case 6:
 document.write("Super Saturday");
 break;
case 0:
 document.write("Sleepy Sunday");
 break;
default:
 document.write("I'm really looking forward to this weekend!");
}
</script>
<p>This JavaScript will generate a different greeting based on what day it is. Note that Sunday=0, Monday=1, Tuesday=2, etc.</p>
</body>
</html>
```

# while loop

- The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is **true**.
- Once the expression becomes **false**, the loop terminates.



# while loop example

```
<html>
<body>
<table border="5">
<script>
var i=1;
while (i<=6)
{
 document.write("<tr>");
 document.write("<td>col 1 row " + i + "</td>")
 document.write("<td>col 2 row " + i + "</td>");
 document.write("</tr>");
 i++;
}
</script>
</table>
</body>
</html>
```

output

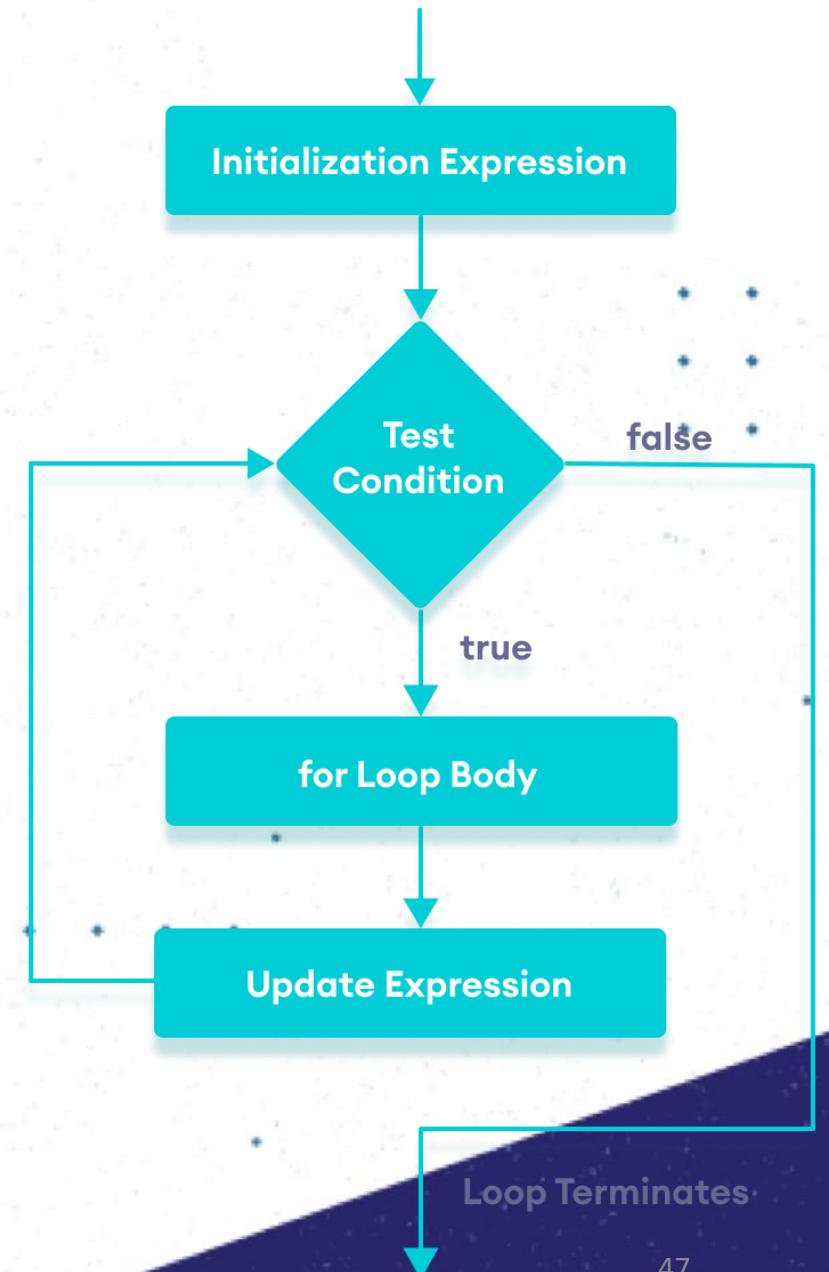
col 1 row 1	col 2 row1
col 1 row 2	col 2 row2
col 1 row 3	col 2 row3
col 1 row 4	col 2 row4
col 1 row 5	col 2 row5
col 1 row 6	col 2 row6

.....

# for loop

```
for (var=startvalue; var<=endvalue; var=var+increment)
```

```
{
 //code to be executed
}
```



# for loop example

```
<html>
<body>
<table border="5">
<script>
 for (i=0;i<=6;i++)
 {
 document.write("<tr>");
 document.write("<td>col 1 row " + i + "</td>")
 document.write("<td>col 2 row " + i + "</td>");
 document.write("</tr>");
 //i++;
 }
</script>
</table>
</body>
</html>
```

output

col 1 row 0	col 2 row 0
col 1 row 1	col 2 row 1
col 1 row 2	col 2 row 2
col 1 row 3	col 2 row 3
col 1 row 4	col 2 row 4
col 1 row 5	col 2 row 5
col 1 row 6	col 2 row 6

## • The break Statement

- The break statement will break the loop and continue executing the code that follows the loop (if any).

## • The continue Statement

- The continue statement will break the current loop and continue with the next iteration.

# Break statement example

```
<html>
 <body>
 <script>
 for (i=0;i<=10;i++){
 if (i==8)
 {
 break;
 }
 document.write("The number is " + i);
 document.write("
");
 }
 document.write("Break....");
 </script>
 </body>
</html>
```

## output

The number is 0  
The number is 1  
The number is 2  
The number is 3  
The number is 4  
The number is 5  
The number is 6  
The number is 7  
Break....

# Continue statement example

```
<!DOCTYPE html>

<html>

<body>

<script>

var i=0

for (i=0;i<=10;i++)
{
 if (i==3)
 {
 continue;
 }
 document.write("The number is " + i);
 document.write("
");
}
</script>
</body>
</html>
```

## output

The number is 0  
The number is 1  
The number is 2  
The number is 4  
The number is 5  
The number is 6  
The number is 7  
The number is 8  
The number is 9  
The number is 10

# Summary

- Introduction to the JavaScript
  - Variables in JS
  - Operators in JS
  - Control structures in JS



IT1100 - Internet and Web Technologies

# Lecture 05

## JavaScript – Part II

# Content

- JavaScript Strings
- Arrays in JS
- DOM API
- Event handling



# JavaScript Strings

# JavaScript Strings

- JavaScript strings are used for storing and manipulating text.
- zero or more characters written inside quotes , using single or double quote.

- var Description= "IWT"

- var Description = 'Lecture 05'

- var Description = 'He is called "Mahela"'

- You can use quotes inside a string, as long as they don't match the quotes surrounding the String

## *Length of a String*

- var txt = "lets watch legend playing";

- var Length=txt.length;

# Escape Character

- Since strings must be written within quotes, JavaScript will misunderstand the following string:

```
var s1 = "IWT Lecture 05 "JavaScript" Part 2.";
var s2 = "IWT Lecture 05 'JavaScript' Part 2";
var s3 = "IWT Lecture 05 \JavaScript\ Part 2";
.....
.....
.....
```

To avoid that problem, use the backslash escape character.

```
var s1 = "IWT Lecture 05 \"JavaScript\" Part 2";
var s2 = "IWT Lecture 05 \'JavaScript\' Part 2";
var s3 = "IWT Lecture 05 \\JavaScript\\ Part 2";
.....
.....
.....
```

# String Search Methods

- The `indexOf()` method returns the index (the position) of the first occurrence

```
var text = "this lecture is JavaScript lecture";
text.indexOf("lecture")
```

- The `lastIndexOf()` method returns the index of the last occurrence of a specified text in a string

```
var text = "this lecture is JavaScript lecture";
text.lastIndexOf("lecture")
```

Both methods return -1 if the text is not found

# Converting Variables to Numbers

- The **Number()** method returns a number converted from its argument.

```
Number("10") // returns 10
Number("IWT") // returns NaN
```

- The **parseInt()** parses a string and returns a whole number.

```
parseInt("10"); // returns 10
parseInt("10.33"); // returns 10
```

# Arrays in JS

# JavaScript Arrays

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

- `var array_name = [item1, item2, ...];`
- `var cars = ["Toyota", "Volvo"];`

# How to insert new elements

- Cars[2] = “BMW”;

# How to display element ant it's value

- `Document.write(Cars[0]);`

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p>The best way to loop through an array is using a standard for loop:</p>
<p id="demo"></p>
<script>
var fruits, text, fLen, i;

fruits = ["Banana", "Orange", "Apple", "Mango"];

fLen = fruits.length;

for (i = 0; i < fLen; i++) {
 document.write(fruits[i]+"/br>");
}
</script>
</body>
</html>
```

## JavaScript Arrays

The best way to loop through an array is using a standard for loop:

Banana  
Orange  
Apple  
Mango

output

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p>The best way to loop through an array is using a standard for loop:</p>
<p id="demo"></p>
<script>
var fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "";
for (i = 0; i < fLen; i++) {
 text += "" + fruits[i] + "";
}
text += "";
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

## JavaScript Arrays

The best way to loop through an array is using a standard for loop:

- Banana
- Orange
- Apple
- Mango

output

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript For/In Loop</h2>
<p>The for/in statement loops through the properties of an object.</p>
<script>
var txt = "";
var person = ["John","Doe","James"];
var x;
for (x in person) {
 txt = txt + person[x] + " ";
}
document.write(txt);
</script>
</body>
</html>
```

## JavaScript For/In Loop

The for/in statement loops through the properties of an object.

John Doe James

output

# Functions in JS

# Functions

- A function is a group of reusable code which can be called anywhere in your program.
- This eliminates the need of writing the same code again and again.
- It helps programmers in writing modular codes.
- Functions allow a programmer to divide a big program into a number of small and manageable functions.

# Function Definition

- Before we use a function, we need to define it.
  - The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

```
<script>
function function_name (parameter-list)
{
 statement(s)
}
</script>
```

# Calling a Function

- To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the code.

```
<html>
<head>
<script>
function sayHello()
{
 document.write ("Hello there!");
}
</script>
</head>
<body>
<script>
 sayHello();
</script>
</body>
</html>
```

output

Hello there!

# Function Parameters

- Till now, we have seen functions without parameters.
- But there is a facility to pass different parameters while calling a function.
- These passed parameters can be captured inside the function
- Any manipulation can be done over those parameters.
- A function can take multiple parameters separated by comma.

# Function Parameters Example

```
<html>
<head>
<script type="text/javascript">
function sayHello(name, age)
{
 document.write (name + " is " + age + " years old.");
}
</script></script>
</head>
<body>
<script>
 sayHello('Zara', 7);
</script>
</body>
</html>
```

output

Zara is 7 years old.

# The return Statement

- A JavaScript function can have an optional **return** statement.
- This is required if you want to return a value from a function.
- **This statement should be the last statement in a function.**
- For *example*, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

# The return Statement

```
<html>
<head>
<script>
function concatenate(first, last)
{
var full;
full = first + last;
return full;
}
function secondFunction()
{
var result;
result = concatenate('Zara ', 'Ali Khan');
document.write (result);
}
</script>
</head>
<body>
<script>
secondFunction();
</script>
</body>
</html>
```

output

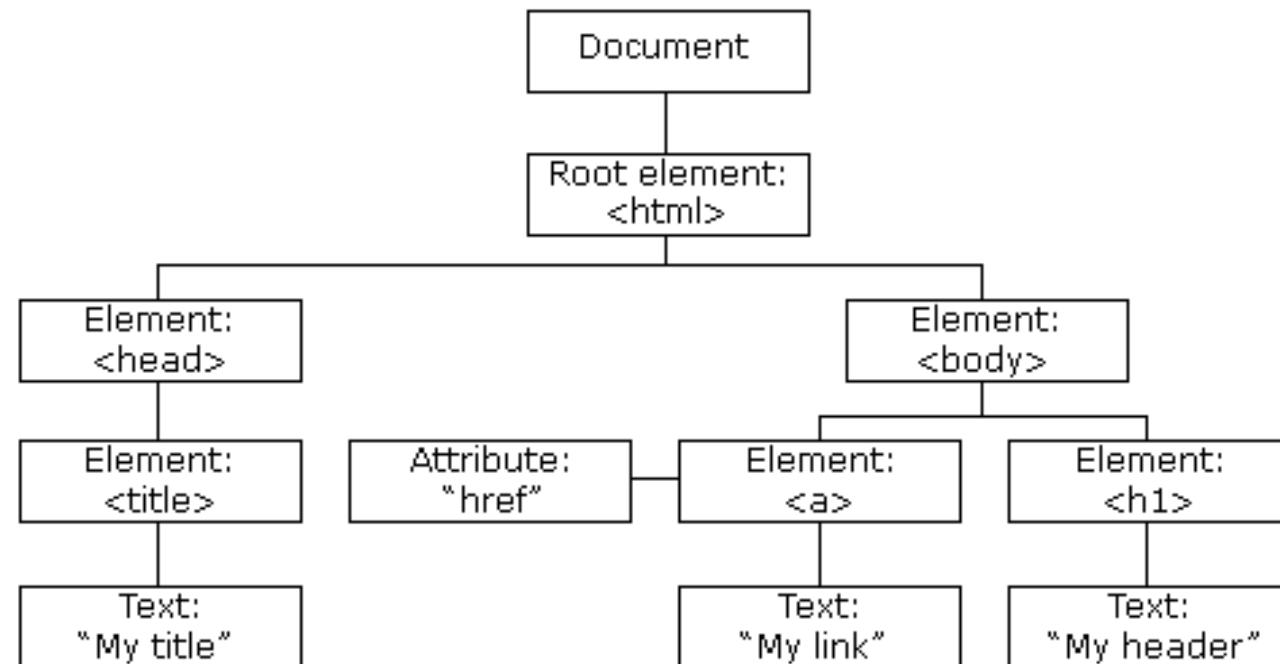
Zara Ali Khan

# Document Object Model

# Document Object Model

When a web page is loaded, the browser creates a Document Object Model of the page.

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

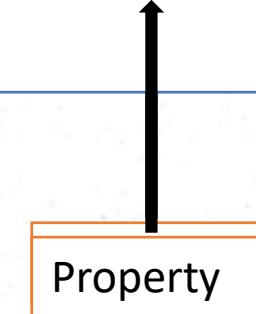
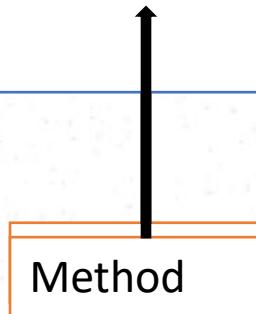


# Document Object Model

In the DOM, all HTML elements are defined as objects.

The programming interface is the properties and methods of each object.

```
<p id="para1"></p>
<script>
document.getElementById("para1").innerHTML = "Hello World!";
</script>
```



# DOM Methods

Method	Description
close()	Closes the output stream previously opened with <code>document.open()</code>
getElementsByName()	Accesses all elements with a specified name
getElementById()	Accesses the element with the specified id
getElementsByClassName()	Accesses all elements with a specified class name
getElementsByTagName()	Accesses all elements with a specified tag name
open()	Opens an output stream to collect the output from <code>document.write()</code> or <code>document.writeln()</code>
write()	Writes HTML expressions or JavaScript code to a document
writeln()	Same as <code>write()</code> , but adds a newline character after each statement

# DOM API

```
<form>
 <input type="text" id="txtName">
 <div id="divOutput"></div>
</form>

//Read the value
var name = document.getElementById("txtName").value;
...
//Display output
document.getElementById("divOutput").innerHTML = "Hello "+name;
```

# Event Handling

# Event Handling

- Event handling is used to implement responses for the user events
  - Click, type, select, drag and drop, etc...

Ex:

- Read form values and validate before submitting the form and display proper error messages

# Event Handling

- Event handlers are used to handle the events, when the events are triggered
- There are 2 main ways of developing event handlers in JS
  - 1. DOM level 0 inline event handlers
  - 2. Event registration using the **addEventListener()** function

# DOM level 0 inline event handlers

- DOM allows to assign events to HTML elements using JavaScript:

- HTML event attributes are used.
  - `onclick`, `onload`, etc...

```
<button onclick="alert('Hello');">Try it</button>
```

# DOM level 0 inline event handlers

- If there is more code to write, it is good to implement a function and call that function in the event handler

```
.....
.....
<button onclick="myFunction();">Try it</button>
.....
.....
<script>
function myFunction() {
 alert("Do whatever needed in this function");
}
.....
</script>
```

# DOM level 0 inline event handlers

```
<html>
<head>
<script>
function myFunction() {
 document.getElementById("demo").innerHTML = document.getElementById("inTxt").value;
}
</script>
</head>
<body>

<p id="demo"></p>
<input type="text" id="inTxt">

<button onclick="myFunction()">Click Me</button>
</body>
</html>
```

The diagram illustrates the DOM structure corresponding to the provided HTML and JavaScript code. It features a light blue background with a grid of small blue dots. Two specific nodes are highlighted with colored boxes and arrows pointing to them in the code:

- A teal box surrounds the element `document.getElementById("demo")`. An arrow points from this box to the `<p id="demo"></p>` line in the HTML.
- A green box surrounds the element `document.getElementById("inTxt")`. An arrow points from this box to the `<input type="text" id="inTxt">` line in the HTML.

Other parts of the code, such as the script block and the button, are shown without highlighting or arrows.

# Event registration using addEventListener()

- It is good to separate the JS from HTML as much as possible, towards increasing the modifiability.
- By using the **addEventListener()** function, we can eliminate the HTML event attributes

# Event registration using addEventListener()

```
<button id="btnTest">Try it</button>
<script>
var btn = document.getElementById("btnTest");
btn.addEventListener("click", function() {
 alert("Do whatever needed in this function");
});
</script>
```

# Event registration using addEventListener()

```
<html>
<head>
</head>
<body>
<input type="text" id="inTxt">

<button id="myBtn">Click Me</button>

<p id="demo"></p>
<script>
document.getElementById("myBtn").addEventListener("click", function(){
 document.getElementById("demo").innerHTML = document.getElementById("inTxt").value;
});
</script>
</body>
</html>
```

# Summary

- JavaScript Arrays
- String and Numerical methods
- DOM API
- Event handling

# Lecture 06

# PHP - Introduction

IT1100 Internet and Web technologies

# Content

- Introduction
- Variables and Constants
- Operators
- Control structures

# Introduction

- PHP is a scripting language for developing server-side components
- The components developed with PHP should be hosted in a compatible web server
  - Apache, IIS

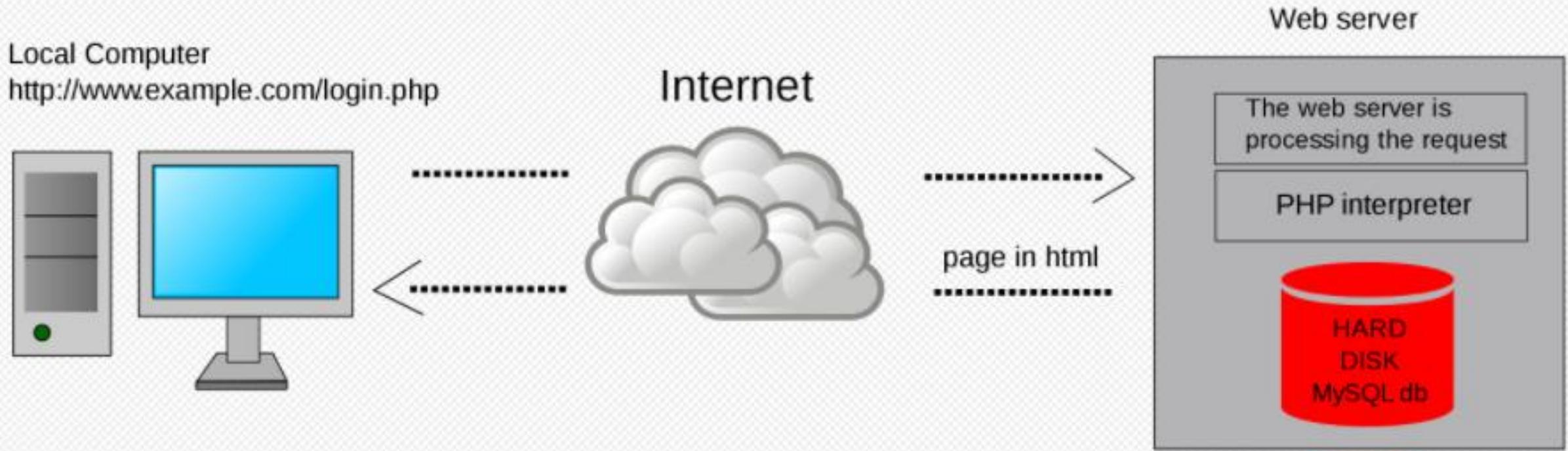
**NOTE:** You will learn to host PHP application and access it, in the practical class.

# What Can PHP Do?

- Generate dynamic page content
- Create, open, read, write, delete, and close files on the server
- Collect form data
- Send and receive cookies
- Insert, delete, update or search data in your database
- Restrict users to access some pages on your website
- Encrypt data
- You can output
  - images, PDF files, Flash movies, text, XHTML and XML.

# How to run your first .php file

1. Write and save php code as a .php file.
2. Copy .php file into the web server.
  - Ex.
    - C:\xampp\htdocs\ita\_demo
3. Open a web browser
4. Type the URL and call your .php file
  - Ex
    - [http://localhost/ita\\_demo/lec\\_1/Test.php](http://localhost/ita_demo/lec_1/Test.php)



# PHP Execution flow

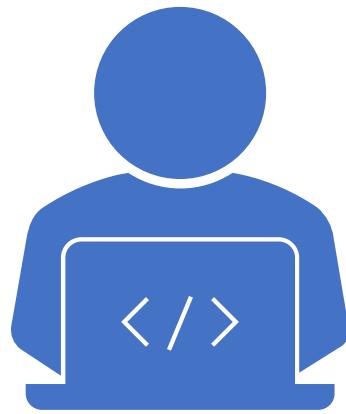
# PHP Execution flow

1. The client requests a page that contains PHP tags.
2. The Web server will pass any page requests containing a PHP file extension to the PHP processor.
3. PHP processor scans the page and processes all PHP tags.  
Might also retrieve information from a database.
4. PHP processor returns only HTML and other client-side technologies to the Web server.
5. The Web server passes the page back to the browser

# PHP Code

- The file/page with PHP script is saved to .php extension
- The script/code is written between the PHP tag

```
<?php
echo "<h1>Hello world</h1>";
// The output may contain HTML
?>
```



# PHP code can be written

---

1. Inline with HTML

//Not recommended

---

2. On the top of the file

//Like internal CSS sheets, JavaScripts

---

3. As an external file

//Like external CSS sheets, JavaScripts

# Inline with HTML

```
<html>
 <head></head>
 <body>
 <h1>PHP example</h1>
 <?php
 echo "<h1>Hello world</h1>";
 ?>
 </body>
</html>
```

## Output

**PHP example**  
**Hello world**

# PHP code on top of the page

```
<?php
 echo "<h1>Hello world</h1>";
?
<html>
 <head></head>
 <body>
 <h1>PHP example</h1>
 </body>
</html>
```

## Output

**Hello world**

**PHP example**

# PHP code in external file

index.php

```
<?php
include("Logic.php"); //Link the external file
?>
<html>
 <head></head>
 <body>
 <h1>PHP example</h1>
 </body>
</html>
```

Output

Hello world

PHP example

Logic.php

```
<?php
echo "<h1>Hello world</h1>";
?>
```

# Variables

- Variable names must begin with a “\$”
- Must not start with a number or special characters ' < & , > ^
- Must not contain spaces.
- Must be less than 32 characters
  - **\$3\_name** – incorrect
  - **\$name\_** – correct
  - **\$name it** – incorrect
  - **\$name** – correct

# Variables

- All user-defined functions, classes, and keywords are NOT case-sensitive.
- All variables are case-sensitive.
- Double quotes ("") will replace a variable's name with its value.
  - \$a=5;
  - echo “\$a”; //Will display 5
- Single quotes ('') will treat them literally (display exactly what you type).
  - \$a=5;
  - echo ‘\$a’; //Will display \$a

# Data types

- PHP is weakly typed language

- string                            \$myString = "Hello world";

- integers                        \$myNumber = 21;

- floating-point /double    \$myNumber = 21.4;

- boolean                        \$gameOver = false;

- How PHP determine the datatype?

# PHP concatenation

- PHP concatenation uses a dot “.”

```
<?php
 $a = "Hello";
 $b = "World";
 $c = $a . " " . $b;
 echo $c;
?>
```

## Output

Hello World

# Double-quoted and Single-quoted strings

- Double quotes can be used within single-quoted strings and vice versa. Both valid:
  - \$phrase = "It's time to go";
  - \$phrase = 'She said "OK"';
- The following are not valid (error due to mismatch of quotes):
  - \$phrase = 'It's time to go';
  - \$phrase = "She said "OK"';
- If you want to use the same quote within a quoted string you must escape it by using a backslash.
  - \$phrase = 'It\'s time to go';
  - \$phrase = "He said \"OK\"";

# Question 1

```
<?php

$variable = 'Saman';
echo 'My name is $variable';

?>
```

**Output**

**My name is \$variable**

```
<?php

$variable = 'Saman';
echo "My name is $variable";

?>
```

**Output**

**My name is Saman**

# Constants

- 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

## Examples:

```
define("UNI", "University of Westminster");
define("PRICE", 79.99);
echo UNI;
print UNI;
```

# Operators - Arithmetic Operators

## PHP Arithmetic Operators

Operator	Name	Example	Result
+	Addition	<code>\$x + \$y</code>	Sum of \$x and \$y
-	Subtraction	<code>\$x - \$y</code>	Difference of \$x and \$y
*	Multiplication	<code>\$x * \$y</code>	Product of \$x and \$y
/	Division	<code>\$x / \$y</code>	Quotient of \$x and \$y
%	Modulus	<code>\$x % \$y</code>	Remainder of \$x divided by \$y

## PHP Increment / Decrement Operators

Operator	Name	Description
<code>++\$x</code>	Pre-increment	Increments \$x by one, then returns \$x
<code>\$x++</code>	Post-increment	Returns \$x, then increments \$x by one
<code>--\$x</code>	Pre-decrement	Decrements \$x by one, then returns \$x
<code>\$x--</code>	Post-decrement	Returns \$x, then decrements \$x by one

# Assignment Operators

Assignment	Same as...	Description
$x = y$	$x = y$	The left operand gets set to the value of the expression on the right
$x += y$	$x = x + y$	Addition
$x -= y$	$x = x - y$	Subtraction
$x *= y$	$x = x * y$	Multiplication
$x /= y$	$x = x / y$	Division
$x \% y$	$x = x \% y$	Modulus

## PHP String Operators

Operator	Name	Example	Result
.	Concatenation	<code>\$txt1 = "Hello" \$txt2 = \$txt1 . " world!"</code>	Now \$txt2 contains "Hello world!"
.=	Concatenation assignment	<code>\$txt1 = "Hello" \$txt1 .= " world!"</code>	Now \$txt1 contains "Hello world!"

# Comparison Operators

## PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
<code>==</code>	Equal	<code>\$x == \$y</code>	True if \$x is equal to \$y
<code>===</code>	Identical	<code>\$x === \$y</code>	True if \$x is equal to \$y, and they are of the same type
<code>!=</code>	Not equal	<code>\$x != \$y</code>	True if \$x is not equal to \$y
<code>&lt;&gt;</code>	Not equal	<code>\$x &lt;&gt; \$y</code>	True if \$x is not equal to \$y
<code>!==</code>	Not identical	<code>\$x !== \$y</code>	True if \$x is not equal to \$y, or they are not of the same type
<code>&gt;</code>	Greater than	<code>\$x &gt; \$y</code>	True if \$x is greater than \$y
<code>&lt;</code>	Less than	<code>\$x &lt; \$y</code>	True if \$x is less than \$y
<code>&gt;=</code>	Greater than or equal to	<code>\$x &gt;= \$y</code>	True if \$x is greater than or equal to \$y
<code>&lt;=</code>	Less than or equal to	<code>\$x &lt;= \$y</code>	True if \$x is less than or equal to \$y

# Logical Operators

## PHP Logical Operators

Operator	Name	Example	Result
and	And	<code>\$x and \$y</code>	True if both <code>\$x</code> and <code>\$y</code> are true
or	Or	<code>\$x or \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true
xor	Xor	<code>\$x xor \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true, but not both
<code>&amp;&amp;</code>	And	<code>\$x &amp;&amp; \$y</code>	True if both <code>\$x</code> and <code>\$y</code> are true
<code>  </code>	Or	<code>\$x    \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true
!	Not	<code>!\$x</code>	True if <code>\$x</code> is not true

[http://www.w3schools.com/php/php\\_operators.asp](http://www.w3schools.com/php/php_operators.asp)

# Array Operators

## PHP Array Operators

The PHP array operators are used to compare arrays:

Operator	Name	Example	Result
+	Union	<code>\$x + \$y</code>	Union of <code>\$x</code> and <code>\$y</code> (but duplicate keys are not overwritten)
==	Equality	<code>\$x == \$y</code>	True if <code>\$x</code> and <code>\$y</code> have the same key/value pairs
===	Identity	<code>\$x === \$y</code>	True if <code>\$x</code> and <code>\$y</code> have the same key/value pairs in the same order and of the same types
!=	Inequality	<code>\$x != \$y</code>	True if <code>\$x</code> is not equal to <code>\$y</code>
<>	Inequality	<code>\$x &lt;&gt; \$y</code>	True if <code>\$x</code> is not equal to <code>\$y</code>
!==	Non-identity	<code>\$x !== \$y</code>	True if <code>\$x</code> is not identical to <code>\$y</code>

[http://www.w3schools.com/php/php\\_operators.asp](http://www.w3schools.com/php/php_operators.asp)

# Control structures

# Selection - simple if-else

```
if ($number < 10)
{
 // code to be executed when the condition is true
 echo "$number is less than ten";
}

else
{
 // code to be executed when the condition is true
 echo "$number is not less than ten";
}
```

# Selection - if-else ladder

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

<?php
$t=date("H");

if ($t<"10") {
 echo "Have a good morning!";
} elseif ($t<"20") {
 echo "Have a good day!";
} else {
 echo "Have a good night!";
}

?>

</body>
</html>
```

Output ?

# Selection - switch

```
<!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,
or green!";
}
?>
</body>
</html>
```

Output ?

Your favorite color is red!

# Question 2

- What is the output ?

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

<?php
$x = rand(1,5); // random integer
echo "x = $x

";
switch ($x)
{
case 1:
 echo "Number 1";
 break;
case 2:
 echo "Number 2";
 break;
case 3:
 echo "Number 3";
 break;
default:
 echo "No number between 1 and 3";
 break;
}
?>

</body>
</html>
```

# Iteration- while

```
while ($i <= 10)
```

```
{
```

```
 echo $i++;
```

```
 $i++;
```

```
}
```

# Iteration- while

- Write a php code to get the following output
  - Use a while loop

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

Output ?

The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5

# Iteration- do while

- Write a php code to get the following output
  - Use a do-while loop

```
<!DOCTYPE html>
<html>
<body>
<?php
$x=4;

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

Output ?

The number is: 4  
The number is: 5

# Iteration - for

```
for ($i = 1; $i <= 10; $i++)
{
 echo $i;
}
```

# Iteration - for

- Write a php code to get the following output
  - Use a for loop

```
<!DOCTYPE html>
<html>
<body>
<?php
for ($x=0; $x<=10; $x++) {
 echo "The number is: $x
";
}
?>
</body>
</html>
```

Output ?

The number is: 0  
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5  
The number is: 6  
The number is: 7  
The number is: 8  
The number is: 9  
The number is: 10

# Iteration - foreach

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

Output ?

red  
green  
blue  
yellow

# Continue and Break

- **Break** ends execution of the current for, foreach, while, do-while or switch structure.
- **Continue** is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and then the beginning of the next iteration.

```
<!DOCTYPE html>
<html>
<body>
<?php
$i = 0;
for ($i = 0;$i <= 5;$i++)
{
if ($i==2)
{
break;
} echo $i;
echo "
";
}
echo "End of for loop" ;
?>
</body>
</html>
```

Output ?  
0  
1  
End of for loop

```
<!DOCTYPE html>
<html>
<body>
<?php
$i = 0;
for ($i = 0;$i <= 5;$i++)
{
if ($i==2)
{
continue;
}
echo $i;
echo "
";
}
echo "End of for loop" ;
?>
</body>
</html>
```

End of for loop

Output ?  
0  
1  
3  
4  
5  
End of for loop

# Summary

- Introduction
- Variables and Constants
- Operators
- Control structures

# Lecture 07

# PHP – Part 2

IT1100 Internet and Web technologies

# Content

- Arrays
- Functions
- Superglobal variables and Form handling

# Arrays

What is an Array?

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

There are two types of arrays in PHP

1. Numeric (indexed) Arrays
2. Associative Arrays

Create an Array in PHP

- In PHP, the `array()` function is used to create an array:  
**array();**

# Numeric (indexed) Arrays

- //Declare array

```
$n=array();
```

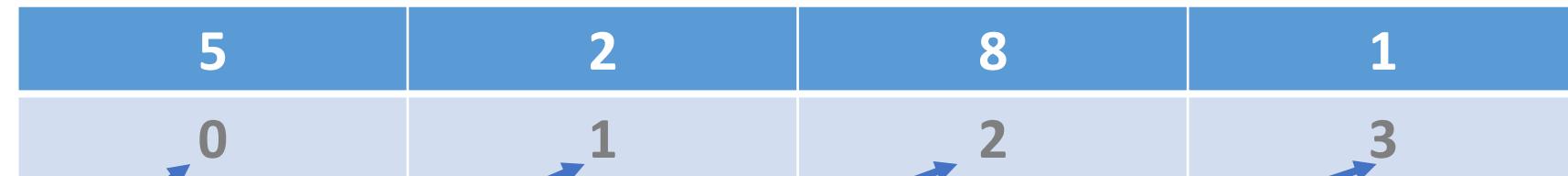
- //Assign values

```
$n[0] = 5;
```

```
$n[1] = 2;
```

```
$n[2] = 8;
```

```
$n[3] = 1;
```



# Numeric (indexed) Arrays

//Declare and initialize array in a single statement

```
$n=array(2,5,8,5);
```

Value	→	2	5	8	5
Index	→	0	1	2	3

- Can read the values of array using the index

```
echo $n[2] + $n[3];
```

2	5	8	5
0	1	2	3

# Loop Through an Indexed Array

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

Declare the array

Loop through the Array

Output

red  
green  
blue  
yellow

The diagram illustrates the execution flow of the PHP code. A red rounded rectangle highlights the entire PHP block. A blue bracket above the opening brace of the foreach loop is labeled 'Declare the array'. Another blue bracket below the closing brace of the foreach loop is labeled 'Loop through the Array'. To the right, a blue box labeled 'Output' contains the four color names: red, green, blue, and yellow.

# Loop Through an Indexed Array

Use a for loop to loop through and print all the values of an indexed array

```
<!DOCTYPE html>
<html>
<body>
<?php
 $subjects=array ("ITA", "DBMS", "ST");
 $arrlength=count ($subjects);
 for ($x=0; $x<$arrlength; $x++) {
 echo $subjects [$x];
 echo "
";
 }
?>
</body>
</html>
```

Declare the array  
Get the array length  
Loop through the Array

Output

ITA  
DBMS  
ST

# Associative Arrays

- For the index
  - Instead of numerical values
  - we can give string IDs for each and every item

```
$marks = array("Maths"=>32, "Web"=>30);
echo $marks["Maths"];
echo $marks["Web"];
```

# Associative Arrays

```
$marks = array();
```

```
$marks["Maths"] = 65;
```

```
$marks["Web"] = 23;
```

```
$marks["JAVA"] = 76;
```

65	23	76
Maths	Web	JAVA

Value →

Key/ID ←

```
echo "Maths: " . $marks["Maths"]. "
".
"Web: " . $marks["Web"]. "
".
"JAVA: " . $marks["JAVA"]. "
";
```

# Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

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

**Output ?**

Key=Peter, Value=35  
Key=Ben, Value=37  
Key=Joe, Value=43

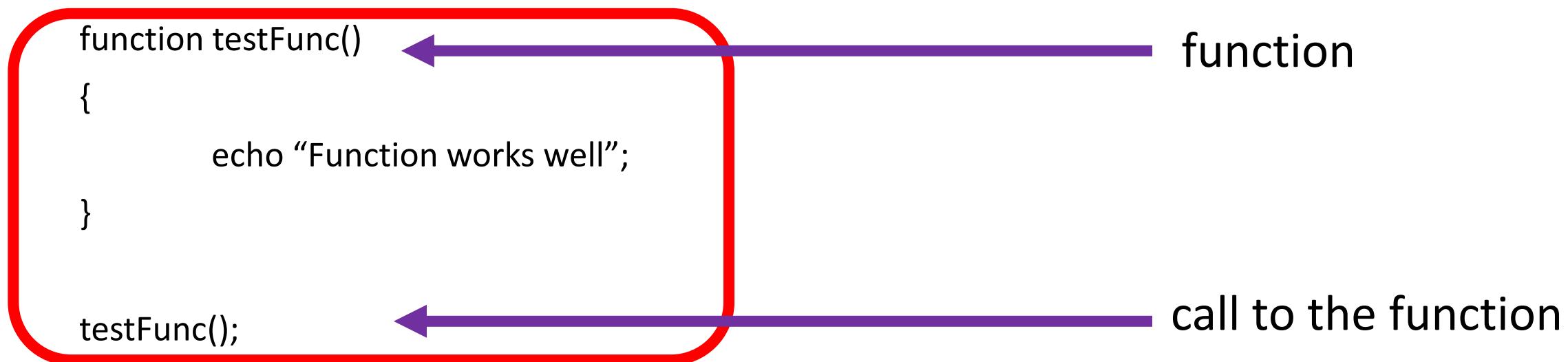


# Functions

- Function is a block of statements that can be used repeatedly in a program.
- PHP Built-in Functions
  - PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.
- PHP User Defined Functions
  - Besides the built-in PHP functions, it is possible to create your own functions.

# User Defined Functions

- function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.



# User Defined Functions

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

<?php
function writeMsg() {
 echo "Hello world!";
}

writeMsg();
?>

</body>
</html>
```



function



call the function

# Global variables

- Subject to the context in which a variable is declared it will have a scope that is either local or global.
- If it is declared within the constraints of a function, it will have local scope and will not be recognized outside of its constraints.
- If you wish to declare a variable with scope beyond its constraints use the term **global**.

# Global variables

```
$name = "Kamal";
$age = 22;

function testFunc()
{
 global $name, $age;
 $greet = "Hello ";
 echo $greet . "Name: " . $name . " - Age: " . $age;
}

testFunc();
```

## Output

Hello Name: Kamal - Age: 22

# User Defined Functions with Parameters

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

<?php
function sum($x, $y) {
 $z = $x + $y;
 return $z;
}

echo "5 + 10 = " . sum(5,10) . "
";
echo "7 + 13 = " . sum(7,13) . "
";
echo "2 + 4 = " . sum(2,4);
?>

</body>
</html>
```

## Output

5 + 10 = 15  
7 + 13 = 20  
2 + 4 = 6

# Superglobal variables

- There are some built-in variables in PHP environment, which provides some special features, they are called **superglobal** variables.
- They are basically associative arrays
- You will learn different types of superglobal variables throughout the module

# Reading forms with Superglobal variables

- HTML form on a web page allows a user to enter data that is sent to a server for processing.

## **`$_GET["index"]`**

- An array of variables passed to the current script via the URL parameters.

## **`$_POST["index"]`**

- An array of variables passed to the current script via the HTTP POST method.

# Reading forms

## Methods to read forms

1. HTML form (frond end) + PHP backend to read form inputs
2. PHP form + PHP backend to read form inputs

# Reading forms (HTML + PHP)

index.html

```
<form method="get" action="process.php">
 Name: <input type="text" name="txtName">

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

process.php

```
<?php
?>
```

```
echo "Hello ". $_GET["txtName"];
```

# Reading forms (HTML + PHP)

## process.php

```
<!DOCTYPE html>

<html>

<body>

<form method="post" action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>>
 Name: <input type="text" name="txtName">

 <input type="submit" value="Submit" name="btnSubmit">

</form>
```

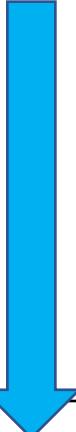
```
<?php
 if(isset($_POST["btnSubmit"])){
 echo "<h1> Hi " . $_POST["txtName"];
 }
?>
```



- **htmlspecialchars** — Convert special characters to HTML entities
- **\$\_SERVER["PHP\_SELF"]** - a super global variable that returns the filename of the currently executing script

# Reading forms - textbox

```
<!DOCTYPE html>
<html>
<body>
<form method="post" action="php-html-form-textbox-example.php">
Name: <input type="text" name="fname">
 <input type="submit">
</form>
</body>
</html>
```



```
<?php
$name = $_REQUEST['fname'];
echo $name;
?>
```

# Reading forms – textarea

```
<html>
<h2>PHP HTML Form textarea Example</h2>
<form name="php-html-form-textarea-example" method="POST" action="php-html-form-textarea-example.php">
 Enter Your Full Name :
 <input name="Fullscreen" type="text" value="">

 Enter Your Address :
 <textarea name="UserAddress" rows="3" cols="20"></textarea>

 <input name="BtnSubmit" type="submit" value="Submit">
</form>
</html>
```

- **isset** — Determine if a variable is declared and is different than NULL

```
<?php
if(isset($_POST['BtnSubmit']))
{
 echo "<h2>Below is the form data</h2>";
 echo "
";
 echo "
Your Name :".$_POST['Fullscreen']."";
 echo "
Your Address :".$_POST['UserAddress']."";
 echo "
";
}
?>
```



## Output

PHP HTML Form textarea Example

Enter Your Full Name : Saman Gamage

New Kandy Road,  
Malabe,  
Sri Lanka

Enter Your Address :  
Submit

Bellow is the form data

Your Name :Saman Gamage

Your Address :New Kandy Road, Malabe, Sri Lanka

# Reading forms - radio

```
<html>
<h3>PHP HTML Form radio button Example</h3>
<form name="UserInformationForm" method="POST" action="php-html-form-radio-button-example.php">
 Enter Your Full Name :
 <input name="FullName" type="text" value="">

 You are :
 <input name="YourGender" type="radio" value="male"> Male
 <input name="YourGender" type="radio" value="female" checked="checked"> Female

 <input name="BtnSubmit" type="submit" value="Submit">
</form>
</html>
```

## Output

### PHP HTML Form radio button Example

Enter Your Full Name :

You are :  Male  Female

Your form data as bellow

Your Name :Saman Gamage  
Your are a:Male

```
<?php
if(isset($_POST['BtnSubmit']))
{
 echo "<h3>Your form data as bellow</h3>";
 echo "
Your Name :{$_POST['FullName']}";
 echo "
Your are :{$_POST['YourGender']}";
 echo "<hr>";
}
?>
```

# Reading forms – checkbox

```
<html>
Check box test
<form action="cbk.php" method="post">
 <input type="checkbox" name="check_list[]" value="one">One

 <input type="checkbox" name="check_list[]" value="two">Two

 <input type="checkbox" name="check_list[]" value="three">Three

 <button type="submit">submit</button>
</form>
</html>
```



```
<?php
echo "Check box test<hr>";
if(!empty($_POST['check_list'])) {
 foreach($_POST['check_list'] as $check) {
 echo "check=$check
";
 }
} else{
}
?>
```

## Output

- empty — Determine whether a variable is empty

Check box test

One  
 Two  
 Three

Check box test

Check list is empty

Check box test

One  
 Two  
 Three

Check box test

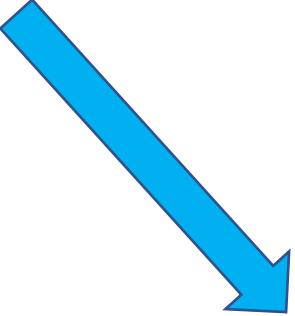
check=one  
check=three

# Reading forms - Select

```
<html>
<h3>PHP HTML Form select box Example</h3>
<form name="UserInformationForm" method="POST" action="php-html-form-select-box-example.php">
 Enter Your Full Name :
 <input name="FullName" type="text" value="">

 I like:
 <select name="I_like">
 <option value="java">JAVA</option>
 <option value="php">PHP</option>
 <option value="asp">ASP .Net</option>
 </select>

 <input name="BtnSubmit" type="submit" value="Submit">
</form>
</html>
```



```
<?php
if(isset($_POST['BtnSubmit']))
{
 echo "<h3>Your submitted form data as bellow</h3>";
 echo "Your Name : {$_POST['FullName']}
";
 echo "I like : <h2>{$_POST['I_like']}</h2> Programming...!
";
 echo "<hr>";
}
?>
```

## Output

<b>PHP HTML Form select box Example</b>	<b>Your submitted form data as bellow</b>
Enter Your Full Name : <input type="text" value="Saman Gamage"/>	Your Name : Saman Gamage
I like: <input type="select" value="ASP .Net"/>	I like :
<input type="button" value="Submit"/>	asp Programming...!

# Summary

- Arrays
- Functions
- Superglobal variables and Form handling

# Lecture 8

# Internet Security

IT1100 – Internet and Web Technologies

# Content

- How safe is the Internet?
- Internet security threats
- Internet security services
- Internet security
- E-life and security

# HOW SAFE IS THE INTERNET?



# How safe is the internet

## eBay e-commerce platform under attack

Share this content:



*A new credit card-stealing attack is underway on the eBay Magento e-commerce platform, which is used by more than 240,000 businesses worldwide.*

The hack has been discovered by US security firm Sucuri, which says it's part of a wave of recent attacks in the wild by the same unnamed hacker group, and predict "we're in for a new trend of Magento-based credit card stealers".

Sucuri senior malware researcher Peter Gramantik said in a [23 June blog](#) that the latest attack exploits a previously unknown vulnerability in the Magento core or one of its widely used modules/extensions.

"Using this vector, the attacker is able to inject malicious code into the Magento core file," he said.



This enables the hacker to intercept 'POST' request from the infected website, giving them all the credit card billing details being sent to the site server.

# How safe is the internet

## Credit card alert as hackers target 77 million PlayStation users

By SEAN POULTER FOR THE DAILY MAIL

UPDATED: 08:06 GMT, 28 April 2011



Share



- Personal data of millions of users worldwide stolen
- Access to PlayStation Network was suspended a week ago, but Sony only revealed details of data theft today
- Fury as Sony announces breach in blog post just hours after launching tablets at high-profile event

Millions of people may be issued with new credit cards over fears their banking details have been stolen by thieves hacking into the Sony PlayStation Network.

The personal information of 77million people around the world is thought to have been compromised.

Some three million Britons – who use the Sony system to play computer games against people in the UK and other countries – have been caught up in the biggest criminal hack on record.

# How safe is the internet And more recently...

## 'Panama papers' came from email server hack at Mossack Fonseca

Money-shuttling firm lost 2.6 TB of data and didn't even notice

5 Apr 2016 at 05:38, Richard Chirgwin



795

The staggering, WikiLeaks-beating "Panama Papers" data exfiltration has been attributed to the breach of an email server last year.

The leak of documents from Panama-based, internationally-franchised firm Mossack Fonseca appears to confirm what has long been suspected but rarely proven: well-heeled politicians, businesses, investors, and criminals use haven-registered businesses to hide their wealth from the public and from taxmen.

*Bloomberg says* co-founder Ramon Fonseca told Panama's Channel 2 the leaked documents are authentic and were "obtained illegally by hackers".

According to *The Spanish*, the whistleblower ([here in Spanish](#)) accessed the vast trove of documents by breaching Mossack Fonseca's email server, with the company sending a message to clients saying it's investigating how the breach happened, and explaining that it's taking "all necessary steps to prevent it happening again".

6

The company added that it's engaged security consultants to close the horse-long-gone stable door.

# How safe is the internet

## A growing threat, Ransomware! 😞



# How safe is the internet

## What & Why Internet Security ?

- Process of creating rules and actions to protect against attacks over the Internet.
- Importance of Internet Security:
  - Privacy & Confidentiality
  - Prevents Data & Identity theft
  - Maintains productivity
  - Foils cyber-terrorism
  - Avoids legal consequences of not securing information
  - And many more

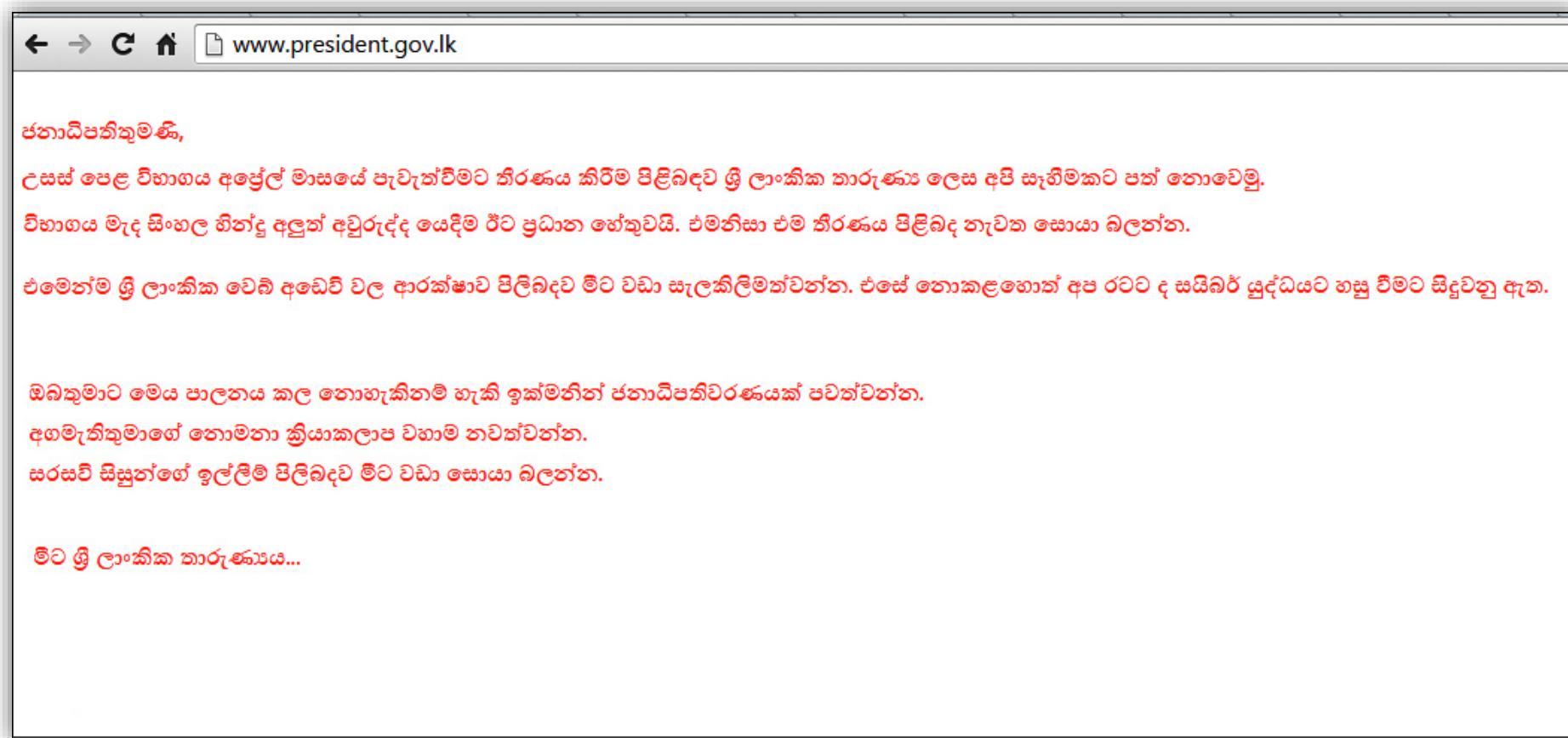
# Internet Security Threats

- Threats are numerous
- Websites are particularly vulnerable
- Political activism is one motivation for Website defacement (Political Espionage)
- Theft of proprietary information is a major concern (Corporate Espionage)
- Some do it for fun, or just to prove a point!

# Internet Security Threats

The screenshot shows the official website for the 2014 FIFA World Cup. At the top, the navigation bar includes links for 'FIFA World Cup™', 'Tournaments', 'World Match Centre', 'FIFA Ranking', and a Google search bar. Below the bar are language links (EN / FR / ES / DE / PT / العربية) and a menu with 'News Centre', 'The Club', 'About FIFA', 'FIFA Ballon d'Or', and 'Associations'. On the right, there are links for 'STORE' and 'Login'. A large banner at the top features the text '\*WE WANT A FAIR WORLD CUP' in green, accompanied by a globe icon and tropical foliage. Below this, a 'SPECIAL MESSAGE: #SAMBAHACK' is displayed. The main content area features a black and white photograph of a man in a white shirt and tie standing on a soccer field, with a large, colorful graphic of a head and shoulders overlaid on his upper body. To the left of the photo is a 'MY TICKETS' button, and to the right is a 'RELIVE THE DRAW EXPERIENCE' section with a 'Match Schedule' PDF link. At the bottom left, a dark banner reads '64 days to go'. The footer contains a summary of the contrast between the 2014 tournament and the first edition in 1930, followed by social media sharing options (Twitter, Google+, Share) and a '10' and '2014 FIFA WORLD CUP' logo.

# Internet Security Threats



# Internet Security Threats

## Types of Attacks

### Denial of Service Attacks

- An explicit attempt by attackers to prevent legitimate users of a service from using that service
- Flooding Attacks
  - Point-to-point attacks: TCP/UDP/ICMP flooding
  - SYN Flood, Ping of Death,
  - Smurf attacks

### Distributed Denial of Service Attacks

- A DoS attack carried out using a large number of compromised systems improving its potency and reducing traceability of the originator.

# Internet Security Threats

## Types of Attacks

### DNS Attacks

- An explicit attempt by attackers to modify a legitimate DNS service.

### Active Code Attacks

- **Java Applets**

- Java: developed by Sun Microsystems
- Java programme runs in browser “sandbox”
- Sandbox might have vulnerabilities
- Problem: hostile applets

- **ActiveX**

- Microsoft’s answer to Java technology
- Web browser can download any type of document
- Internet Explorer invokes handler (example: Word)
- Problem: browser loses control

# Internet Security Threats

## Types of Attacks

### SQL injection

- Attacker execute malicious SQL statements
- Any website or web application that use an SQL-based database could be vulnerable
- Attacker can use this to:
  - Bypass authentication and authorization mechanisms
  - Retrieve the contents of an entire database
  - Add, modify and delete records in a database
- Protection - Need to sanitize user input data

# Internet Security Threats

## Types of Attacks

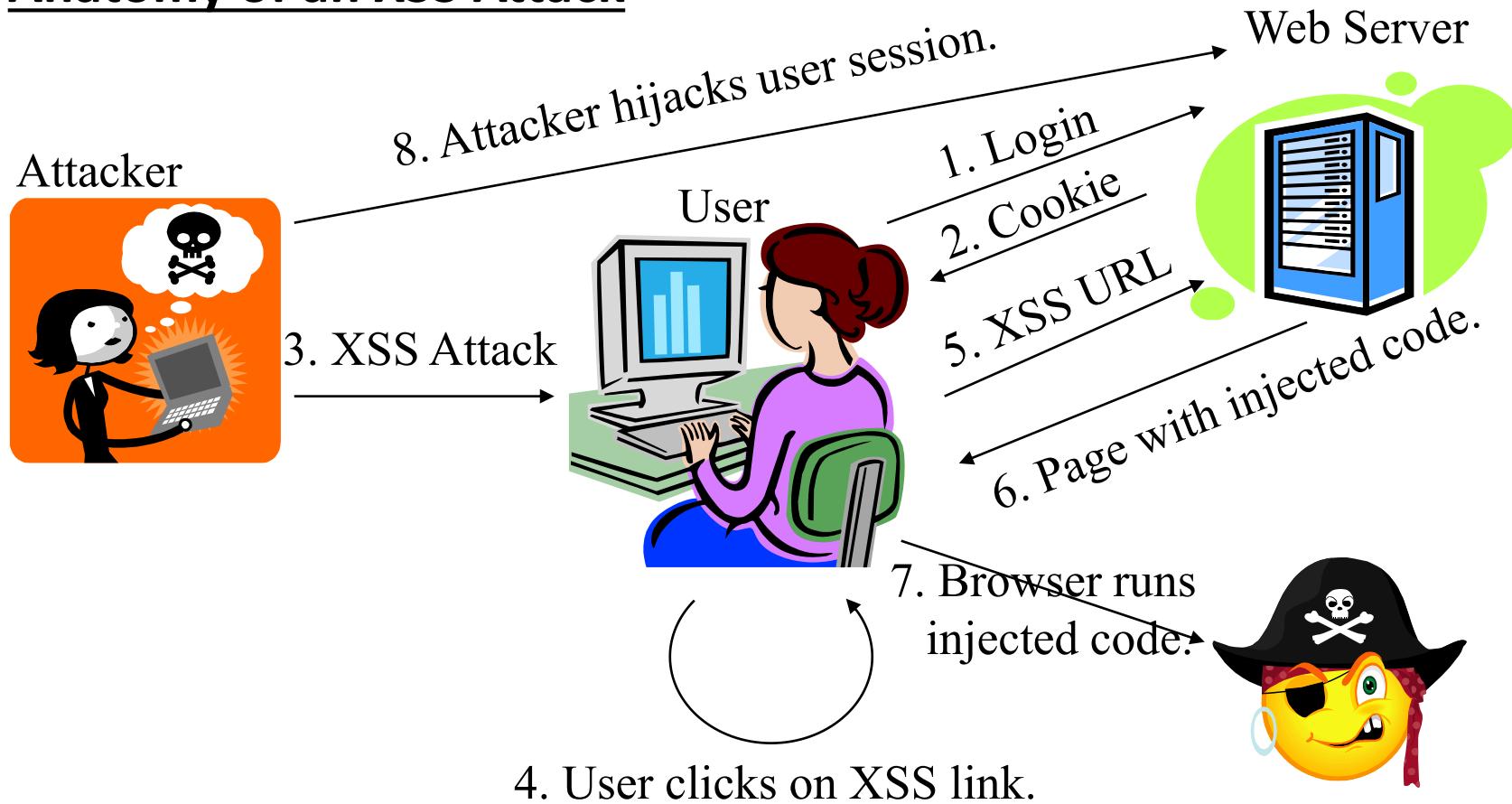
### Cross Site Scripting (XSS)

- Cross Site Scripting (CSS for short, but sometimes abbreviated as XSS) is one of the most common application level attacks that hackers use to sneak into web applications today.
- CSS attack involves three parties – the attacker, a client and the web site.
- The goal of the CSS attack is to steal the client cookies, or any other sensitive information, which can identify the client with the web site. With the token of the legitimate user at hand, the attacker can proceed to act as the user in his/her interaction with the site – specifically, impersonate the user.

# Internet Security Threats

## Types of Attacks

### Anatomy of an XSS Attack



# Internet Security Services

- 1. Confidentiality**
- 2. Integrity**
- 3. Availability**

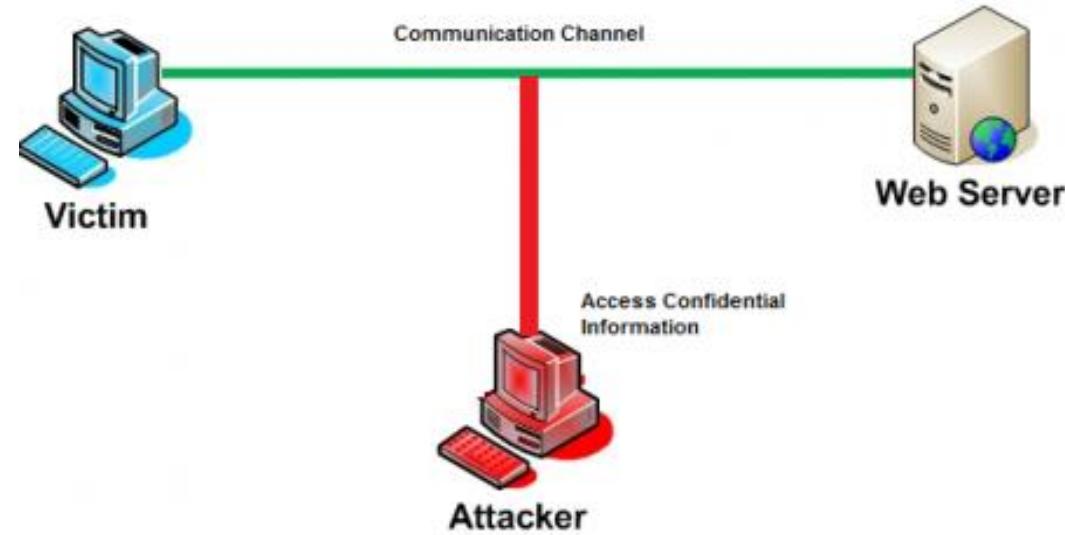
- Non-Repudiation
  - Authentication
  - Authorization (Access Control)
- [Can be described as part of Integrity]**



# Internet Security Services

## Confidentiality

- Can be defined as the '**Secrecy**' of the message
- Also known as '**Privacy**'

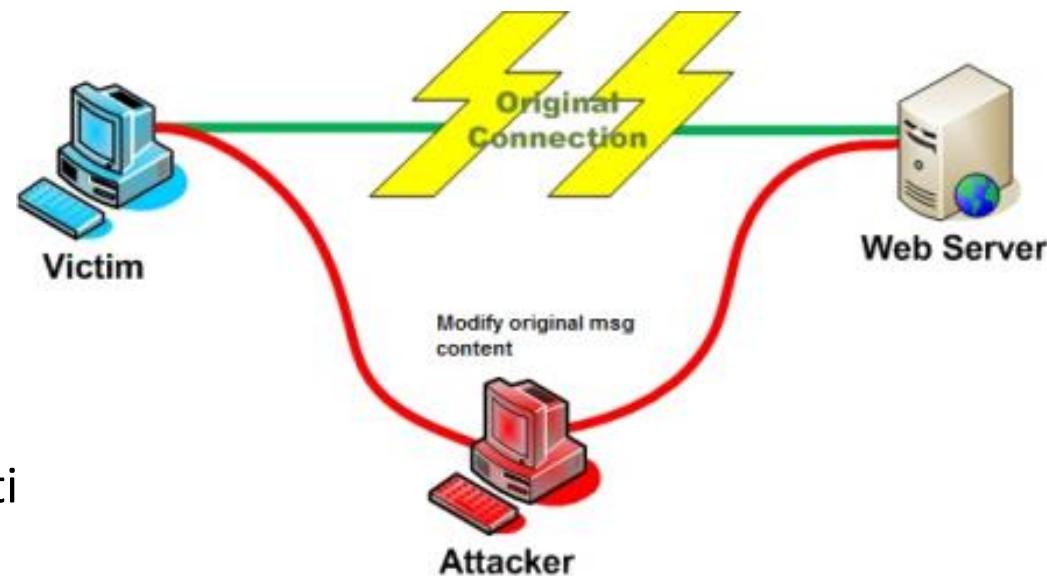


- Note: attacker doesn't change any information (Passive Intruder)

# Internet Security Services

## Integrity

- Can be defined as the '**Security**' of the message
- Ensure that the message will be delivered to the receiver **without being modified** in the middle by the intruder.



- Note: attacker acti  
(Active Intruder)

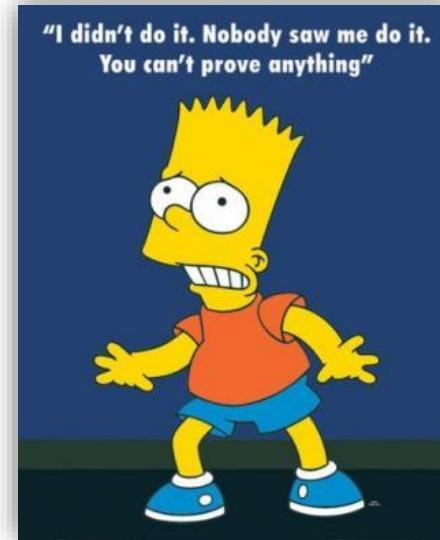
# Internet Security Services Availability

- Assuring information and communications services will be ready for use when expected.
- Information must be kept available to **authorized** persons when they need it.

# Internet Security Services

## Authentication

- Can be called as source verification
- Receiver can identify the actual sender
- Can be defined as a part of Origin Integrity
- Intruder is active and a malicious **impersonator**



## Non – Repudiation

- Sender cannot deny that he/she sent the message
- Can be defined as a part of Origin Integrity



## Authorization (Access Control)

- Access to resources should be in a controlled way.
- Intruders will be active. And try to access the resources which are not allowed.

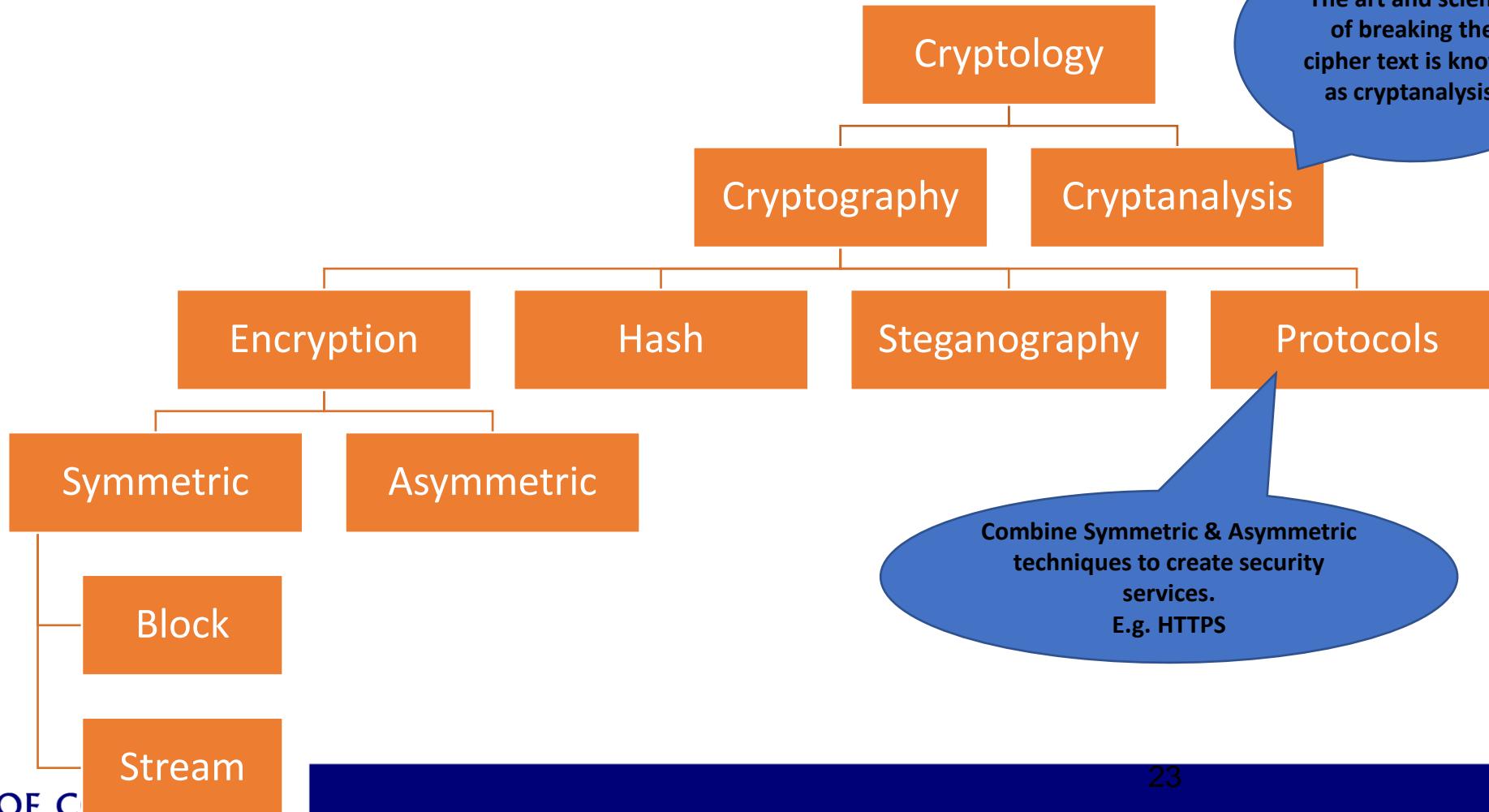
# Internet Security

- **Why is it so difficult to achieve Internet Security?**

- Internet was created as a product of research
- Security was not a prime concern those days
- Inherent protocols does not have security features
  - **HTTP** - Messages sent in plain-text 😞
  - **Email** - No authentication of the sender
  - **FTP** - Basic file transfer protocol has no encryption
- Security had to be introduced as add-ons
  - New protocols had to be introduced  
(**HTTPS, SMTP, SFTP** etc.)
  - But cannot stop users from using old protocols! 😞
- All these new protocols rely on **CRYPTOGRAPHY!**

# Internet Security

## The World of Cryptology

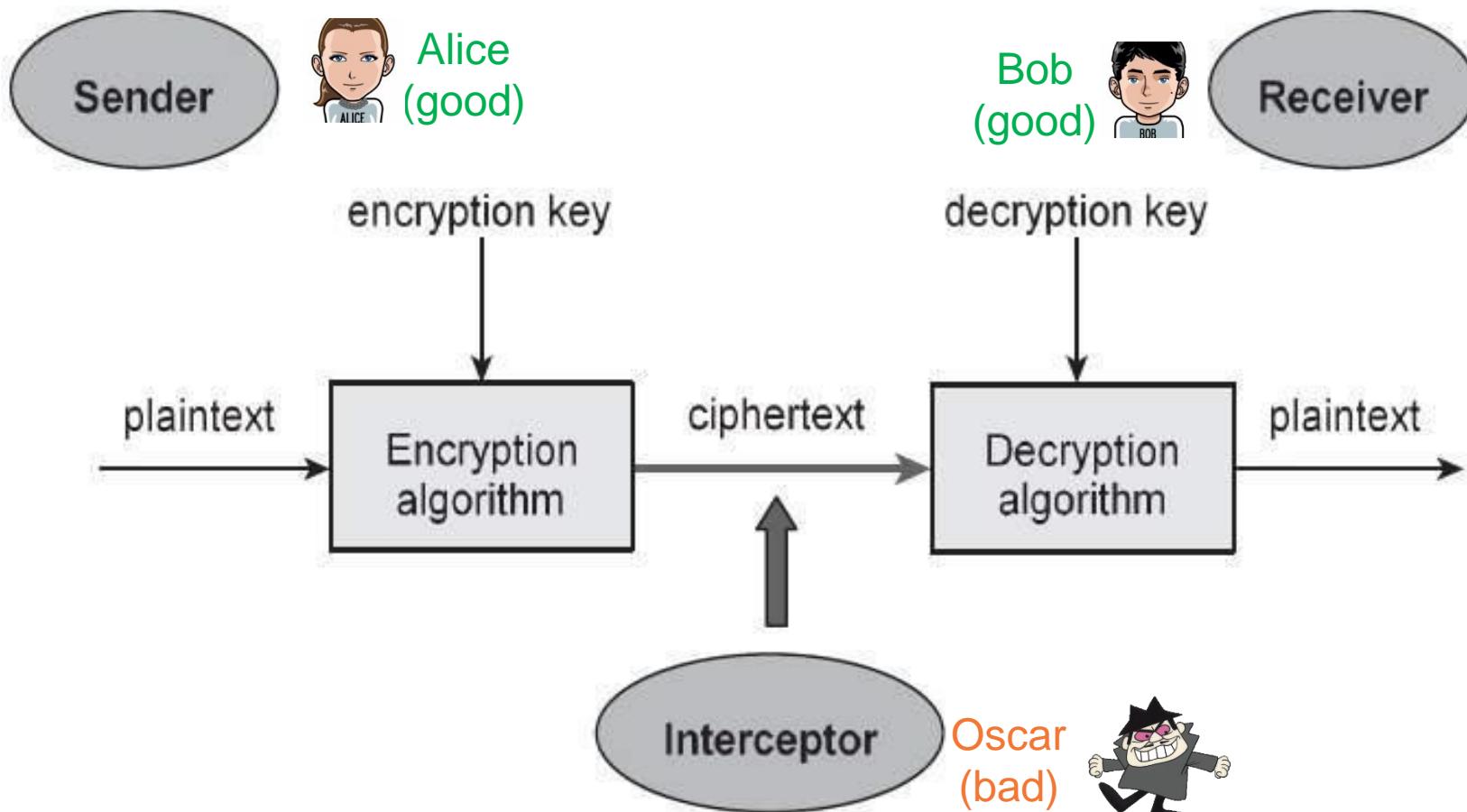


# Internet Security

## Keywords in Cryptography

- **Plain-text:** It is the data to be protected during transmission. Original message
- **Cipher-text:** It is the scrambled version of the plain-text produced by the encryption algorithm using a specific the encryption key. Text in unreadable format
- **Encryption Algorithm:** It is a mathematical process that produces a cipher-text for any given plain-text and encryption key.
- **Decryption Algorithm:** It is a mathematical process, that produces a unique plain-text for any given cipher-text and decryption key.
- **Encryption Key:** It is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plain-text in order to compute the cipher-text.
- **Decryption Key:** It is a value that is known to the receiver. The decryption key is related to the encryption key, but is **not always** identical to it. The receiver inputs the decryption key into the decryption algorithm along with the cipher-text in order to compute the plain-text.

# Internet Security Encryption



# Internet Security Encryption

- openSSL can be used to encrypt/decrypt
- Key generation
  - `base64_encode(openssl_random_pseudo_bytes(32));`
- Encryption
  - Algorithm: 3DES, AES(Rijndael)
  - Modes: ECB, CBC

# Internet Security Encryption

```
function encrypt($plaintext)
{
 $key = pack('H*',
 "bcb04b7e103a0cd8b54763051cef08bc55abe029fdebae5e1d417e2ffb2a00a3");
 $iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC);
 $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
 $ciphertext = mcrypt_encrypt(MCRYPT_RIJNDAEL_128, $key, $plaintext,
 MCRYPT_MODE_CBC, $iv);
 $ciphertext = $iv . $ciphertext;
 $ciphertext_base64 = base64_encode($ciphertext);

 return $ciphertext_base64;
}
```

# Internet Security

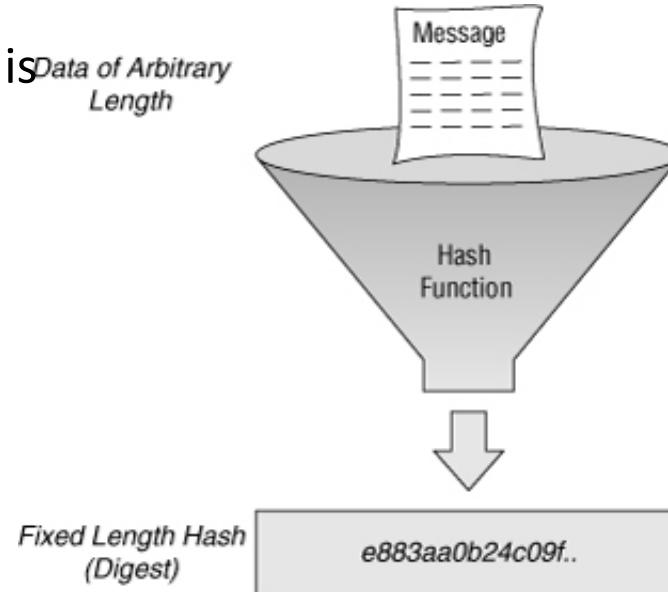
## Decryption

```
function decrypt($cText)
{
 $ciphertext_dec = base64_decode($cText);
 $iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC);
 $iv_dec = substr($ciphertext_dec, 0, $iv_size);
 $ciphertext_dec = substr($ciphertext_dec, $iv_size);
 $key = pack('H*',
 "bcb04b7e103a0cd8b54763051cef08bc55abe029fdebae5e1d417e2ffb2a00a3");
 $plaintext_dec = mcrypt_decrypt(MCRYPT_RIJNDAEL_128, $key, $ciphertext_dec,
 MCRYPT_MODE_CBC, $iv_dec);
 return $plaintext_dec;
}
```

# Internet Security

## Hash Functions

- A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length. E.g. MD5, SHA1, SHA2
- Values returned by a hash function are called message digest or simply hash
  - (160 and 512 bits.)
- Can use for Message Integrity
  - Ideal for Password storage



# Internet Security

## Hash

- Cryptographic libraries can be used to secure the password using Hashing
- Hashing //Store the password's hash into the DB
  - `password_hash($pw, PASSWORD_BCRYPT);`
- Verify the password
  - `password_verify($pw, $hashedPW);`



Stored in  
the DB

# Encryption Algorithms

- **Symmetric key cryptography**
  - Use of a single key for encrypting and decrypting the information
  - **Key exchange problem!**
  - Encryption, decryption algorithms are less complex
  - **Faster!**
- **Asymmetric key cryptography (Public Key)**
  - Each person has their own pair of keys,
    - private key
    - public key (Verified Certificate Authority)
  - Use one key to encrypt and another to decrypt.
  - **Key exchange problem solved!**
  - Encryption, decryption algorithms are very complex
  - Require far greater computation time, **Slower!**
  - impractical for large messages

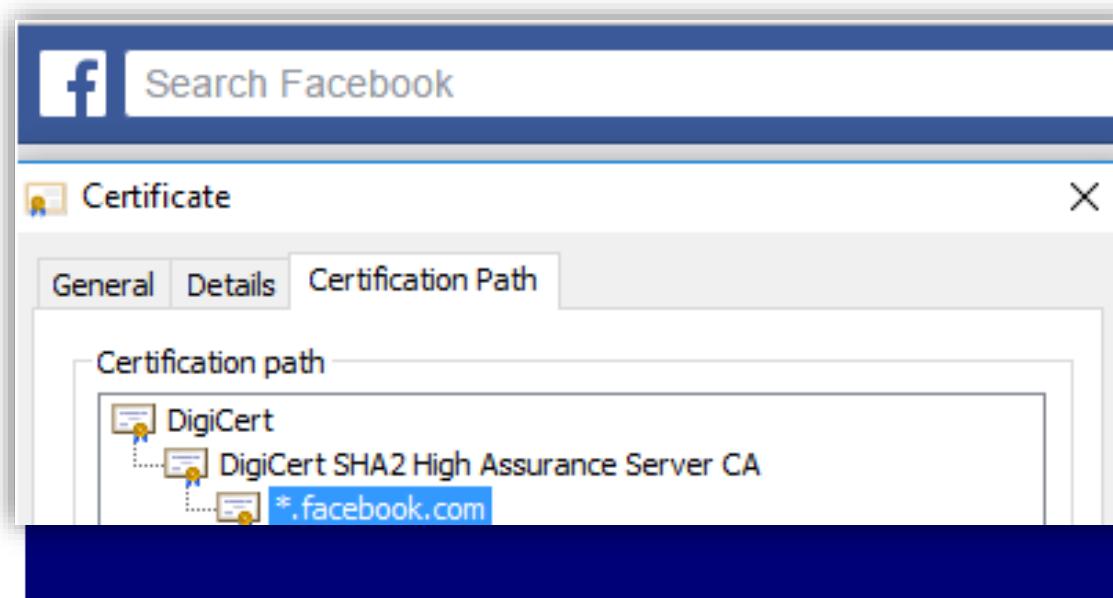
# Certification Authorities - CA

## What CA does ?

- Allow an individual to verify another person or company's public key
- The digital certificate is encrypted with the CA's private key.
- Each person must know the public key for the CA in order to verify the certificate
- Public keys for the main CA are encoded in latest versions of web browsers.

# Certification Authorities - CA

- VeriSign - [www.verisign.com](http://www.verisign.com)
- GeoTrust - [www.geotrust.com](http://www.geotrust.com)
- Comodo - [www.comodo.com](http://www.comodo.com)
- DigiCert - [www.digicert.com](http://www.digicert.com)
- Thawte - [www.thawte.com](http://www.thawte.com)
- GoDaddy - [www.godaddy.com](http://www.godaddy.com)
- Symantec - [www.symantec.com](http://www.symantec.com)



# Internet Security Protocols

- We can combine all the learnt techniques to create comprehensive security protocols.
- Secure Socket Layer (**SSL**) also known as Transport Layer Security (**TLS**)
- Secure IP Protocol (**IPSec**) signature helps to assure that the signer is who he or she claims to be.
- **HTTPS** (Conventional HTTP used inside TLS)
- And many more

# Internet Security Protocols – SSL/TLS

- Provide authentication for servers and browsers.
- Provide confidentiality and data integrity for communications.
- Operating lower in the network stack.
- Widely used on the web.
- SSL makes use of a combination of public key and symmetric cryptography, along with digital certificates for verification.
- Latest version SSL 3.0, TLS 1.2.

# Internet Security Protocols – SSL/TLS



# e-Life & Security



## Best Security Practices - Facebook



- Take control of your posts
  - Understand the importance of FB's **Audience Selector** tool.
- Manage posts other people have tagged you in using **Tag Review**.
- Give priority to your privacy.
  - "Who can see my stuff?" "Who can contact me?" "Who can look me up?"
- Protect your **password!**
- Avoid clicking on suspicious links or objects
- If required you **CAN** permanently delete your FB account.
- Use FB's security guidelines:  
<https://www.facebook.com/facebookmedia/best-practices/security>

## Best Security Practices – YouTube



- Keep your account recovery information updated
- **Audience Selector** in YouTube.
- Videos violating your privacy ?
  - If you can't reach an agreement with the uploader, or if you are uncomfortable contacting them, you can **request removal of content**.
- Unique & strong password!
- Don't be a victim of phishing attacks:
  - YouTube will never ask you for your password, email address, or other account information.
  - Don't be fooled if someone contacts you pretending to be YouTube!

## Best Security Practices – Gmail



- Constantly check recent account activity for anything suspicious.  
Use '**Recent Security Event**' tool.
- Enable '**Authentication icon for verified senders**' tool.
- Do not open 'Spam' emails unless 100% certain
- Unique & strong password!
- Update your web browser
- Complete Gmail's **Security Checkup**

# e-Life & Security

## Lessons Learned

- Internet is **NOT** safe, hence security techniques had to be introduced.
- Many threats exist and attackers use various techniques.
- Key Internet Security services are identified (CIA Triad).
- Cryptography provides solution for security requirements.
- We can combine cryptographic techniques to create comprehensive security protocol suites such as TLS.
- Always use best security practices for your daily internet based interactions.

# Summary

- How safe is the Internet?
- Internet security threats
- Internet security services
- Internet security
- E-life and security

# Lecture 09

# PHP - Database handling

IT1100 Internet and Web technologies

# Content

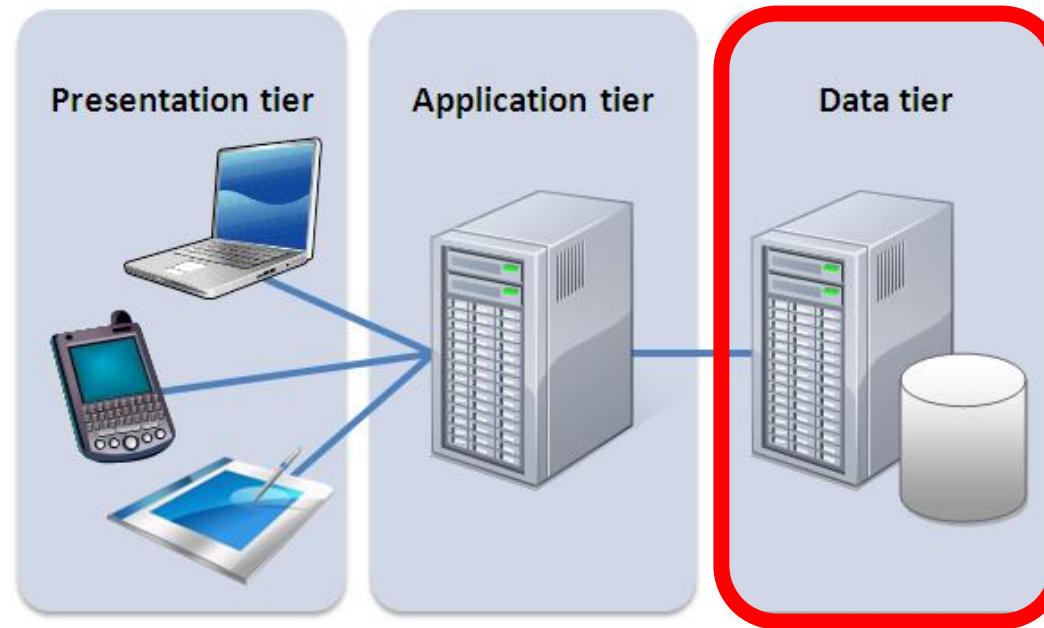
- Introduction
- The connection
- **C**reate
- **R**ead
- **U**pdate
- **D**elete

# Introduction

- How to store data in software applications?
- What is the best method to store data in software applications?
- What is a Database?

# Introduction

- **Database** is an external resource, hosted in a **database server**, and managed by a **DBMS**
- The database server is considered as a separate tier



# Introduction

- **MySQL database server** is the de facto standard for PHP applications
- There are multiple ways to connect to a database using PHP
- PHP can perform **CRUD** operations on a database using SQL

# Ways to connect to DB using PHP

## 1. MySQL extension

- Support only PHP versions before v7
- Procedural

## 2. MySQLi (improved)

- Support since PHP version 7
- Support both procedural and OOP
- Support prepared statements

## 3. PHP Data Objects (PDO)

- A lightweight, consistent interface for accessing databases in PHP.
- Support many DB servers
- Only OOP
- Support prepared statements

# The connection - Configurations

- It is a good idea to keep the DB configurations in a dedicated file config.php

```
//The connection object
```

```
//$con= new mysqli("Server", "UN", "PW", "DB");
```

```
$con=new mysqli("localhost","root","123","test");
```

# The connection - Configurations

- Check for errors before continue

```
// Check connection
if ($con->connect_error)
{
 die("Connection failed: " . $con->connect_error);
}
```

The **connect\_error** function returns the error description from the last connection error, if any. NULL if no error occurred.

# The connection - Configurations

- The configuration file can be linked when needed
- index.php (or any other page/file)

```
//Linking the configuration file
require 'config.php';
```

- require is identical to include except upon failure it will also produce a fatal E\_COMPILE\_ERROR level error.
- It will halt the script whereas include only emits a warning (E\_WARNING) which allows the script to continue.
- The require\_once statement is identical to require except PHP will check if the file has already been included, and if so, not include (require) it again

# Create - The INSERT statement

- To create data, an insert SQL statement is used

```
$sql= "INSERT INTO myTable(ID, Name) VALUES (1, 'SLIIT');
```



Single quotes  
for strings,  
within double  
quotes

# Execute the statement

- `$con->query($sql)`

- This returns a Boolean value to indicate the (un)successful execution of the statement in the DB server

# Execute the statement

```
if ($con->query($sql))
{
 echo "Inserted successfully";
}
else
{
 echo "Error: ". $con->error;
}
```

# Execute the statement

- Do not forget to close the connection
  - After executing any operation

```
$con->close();
```

# Complete Code

```
<?php
//Linking the configuration file
require 'config.php';
$sql= "INSERT INTO myTable(ID,Name)VALUES(11111,'SLIIT');
if($con->query($sql)){
 echo "Inserted successfully";
}
else{
 echo "Error:". $con->error;
}
$con->close();
?>
```

```
<?php
//The connection object

$con=new mysqli("localhost","root","","MyDB");
// Check connection
if($con->connect_error){
 die("Connection failed: " . $con->connect_error);
}
?>
```

config.php

# Problems in data INSERT method

- Can insert One Record at a time
- User need access rights to internal .PHP pages stored in webserver (ex. /htdocs/...)

## Solutions

- Use a HTML Form
- Use a PHP Form

# Solution1

## Use a HTML Form

```
<!doctype html>
<html>
 <head> </head>

 <body>
 <form method="post" action="form_process.php">
 <h3>Input Student Data </h3>
 Student ID :<input type="text" name="stuID">

 Student Name :<input type="text" name="stuName">

 <input type="submit" value="Submit">
 <input type="reset" value="Reset">
 </form>
 </body>
</html>
```



```
<?php
//Linking the configuration file
require 'config.php';
$ID = $_POST["stuID"];
$Name = $_POST["stuName"];
$sql= "INSERT INTO myTable(ID,Name)VALUES($ID,$Name)";
if($con->query($sql)){
 echo "Inserted successfully";
}
else{
 echo "Error:". $con->error;
}
$con->close();
?>
```

```
<?php

//Linking the configuration file
require 'config.php';

?>

<form method="post" action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>>
 <h3>Input Student Data </h3>
 Student ID :<input type="text" name="stuID">

 Student Name :<input type="text" name="stuName">

 <input type="submit" value="Submit" name="btnSubmit">
 <input type="reset" value="Reset">
</form>

<?php
if(isset($_POST["btnSubmit"])){
 $stuID = $_POST["stuID"];
 $stuName = $_POST["stuName"];

 $sql= "INSERT INTO myTable(ID,Name)VALUES($stuID,'$stuName')";
 if($con->query($sql)){
 echo "Inserted successfully";
 }
 else{
 echo "Error:". $con->error;
 }
}
$con->close();
?>
```

## Solution 2

### Use a PHP Form

# Select statement

- When reading data from a DB, we use a select statement, which returns a dataset as the result.

```
$sql = "select ID, name from myTable"
```

# Read - Result set

- We execute the select SQL statement, then assign the result set into a variable

```
$result = $con->query($sql);
```

# Read - Result set - availability

- If only there are results, we can read them

```
if ($result->num_rows > 0)
{
 //read data
}
else
{ echo "no results"; }
```

# Result set – read data

- We read the dataset row by row using a loop
- There are multiple functions to fetch a row from a dataset
  - `fetch_all` — Fetches all result rows as an associative array, a numeric array, or both
  - `fetch_array` — Fetch a result row as an associative, a numeric array, or both
  - `fetch_assoc` — Fetch a result row as an associative array
  - `fetch_field_direct` — Fetch meta-data for a single field
  - `fetch_field` — Returns the next field in the result set
  - `fetch_fields` — Returns an array of objects representing the fields in a result set
  - `fetch_object` — Returns the current row of a result set as an object
  - `fetch_row` — Get a result row as an enumerated array

# Result set – read data

- Lets use `fetch_assoc()`, which return the row as an associative array

```
while($row = $result->fetch_assoc())
{
 //Read and utilize the row data
}
```

# Result set – read data

- Column names can be used as the indexes to read the cell data in the fetched row

```
echo $row["ID"]. " – " . $row["Name"] . "
";
```

EX: show the data inside a table, on the page

# Read -Complete function

- Column names can be used as the indexes to read the cell data in the fetched row

```
echo $row["ID"]. " – ". $row["Name"] . "
";
```

EX: show the data inside a table, on the page

```
<?php
//Linking the configuration file
require 'config.php';

$sql = "select ID, Name from myTable";
$result = $con->query($sql);

if($result->num_rows > 0){
 //read data
 while($row = $result->fetch_assoc()){
 //Read and utilize the row data
 echo $row["ID"] . " - " . $row["Name"] . "
";
 }
}
else
{
 echo "no results";
}

$con->close();
?>
```

```
<?php
//The connection object
$con=new mysqli("localhost","root","","","MyDB");
// Check connection
if($con->connect_error){
 die("Connection failed: " . $con->connect_error);
}
?>
```

# Complete Code

# Read Complete function

```
<?php
require 'config.php';
function readData()
{
 global $con;
 $sql = "SELECT ID, Name FROM myTable";
 $result = $con->query($sql);
 if ($result->num_rows > 0)
 {
 while($row = $result->fetch_assoc())
 {
 echo "ID: " . $row["ID"]. " - Name: " . $row["Name"]. "
";
 }
 }
 else
 {
 echo "No results";
 }
 $con->close();
}
readData();
```

```
<?php
$con=new mysqli("localhost","root","","test");
if($con->connect_error)
{
 die("Connection failed: ". $con->connect_error);
}
?>
```

# Read

- Modify the given php code to display data inside a table.

# Summary

---

Introduction

---

The connection

---

Create

---

Read

---

# Lecture 10

# Cookies and Sessions

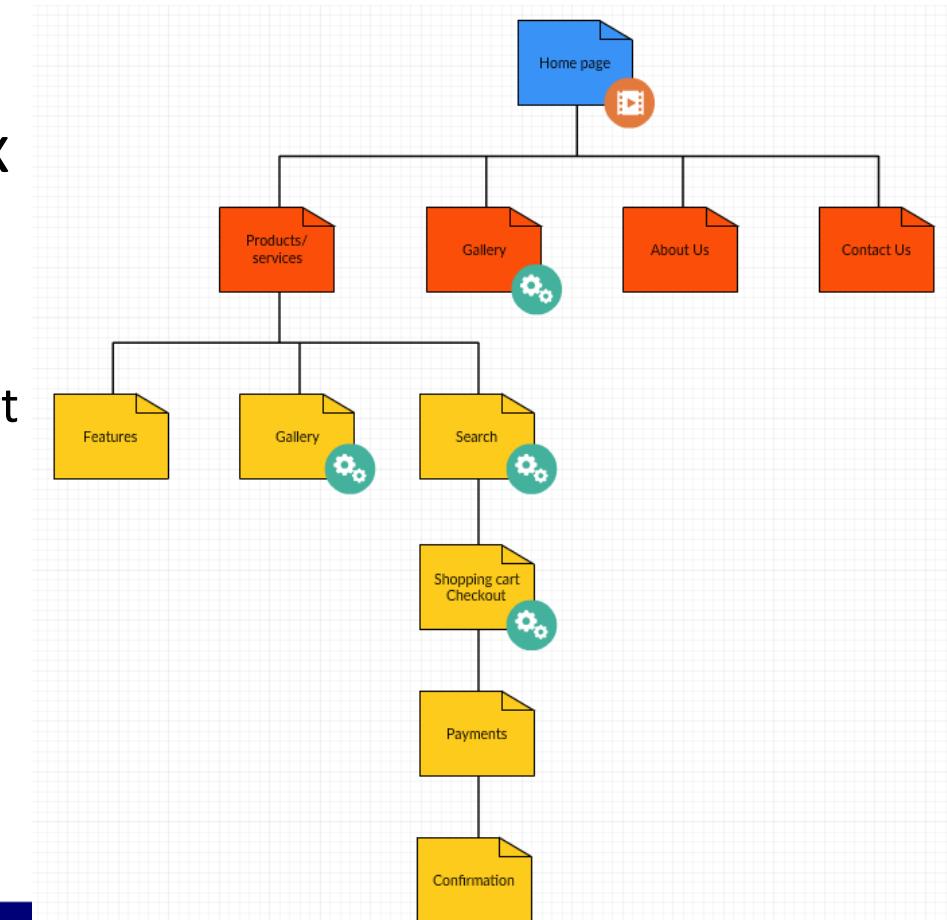
IT1100 Internet and Web technologies

# Content

- Introduction to PHP Cookies and Sessions
- Cookies
- Sessions
- Some use of the sessions

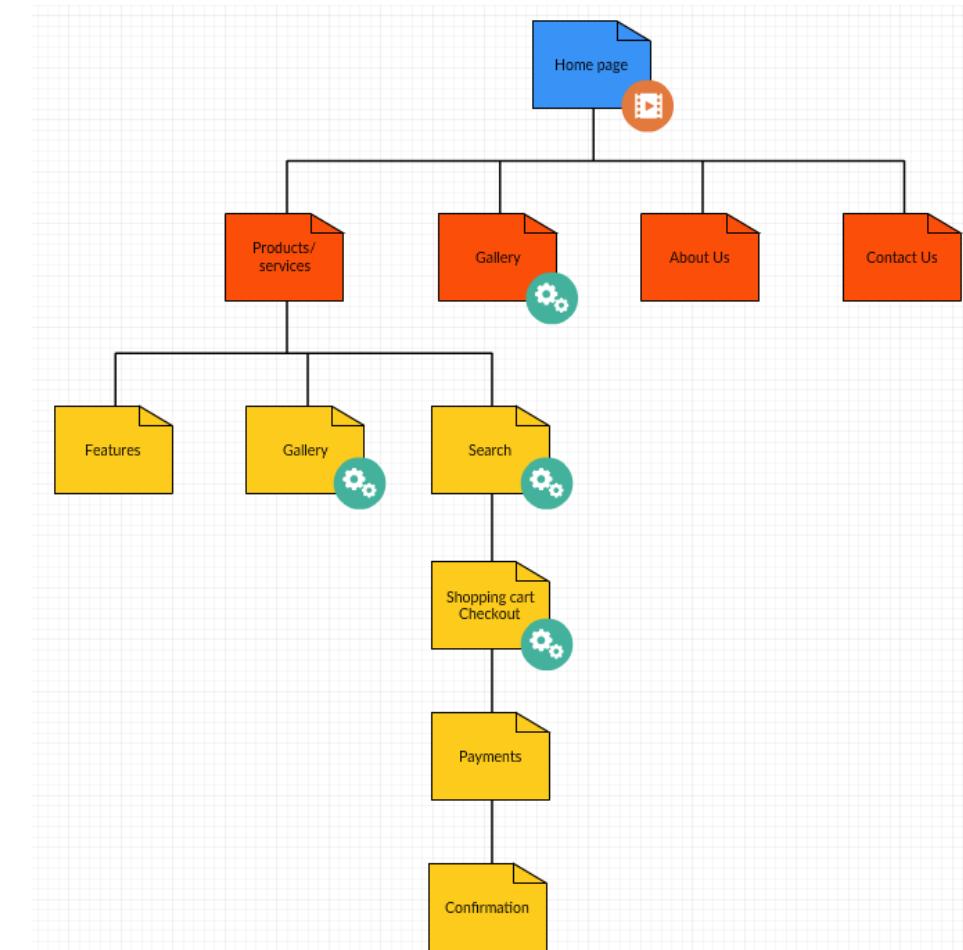
# Introduction to PHP Cookies and Sessions

- A web application may contain multiple pages
  - a page may contain multiple features
- A user may navigate through multiple pages and use multiple features, while performing complex transactions, when accessing a web application
  - This may cause many request-response cycles
  - E.g. – Online shopping application -> items pages, shopping cart page, checkout page, etc...



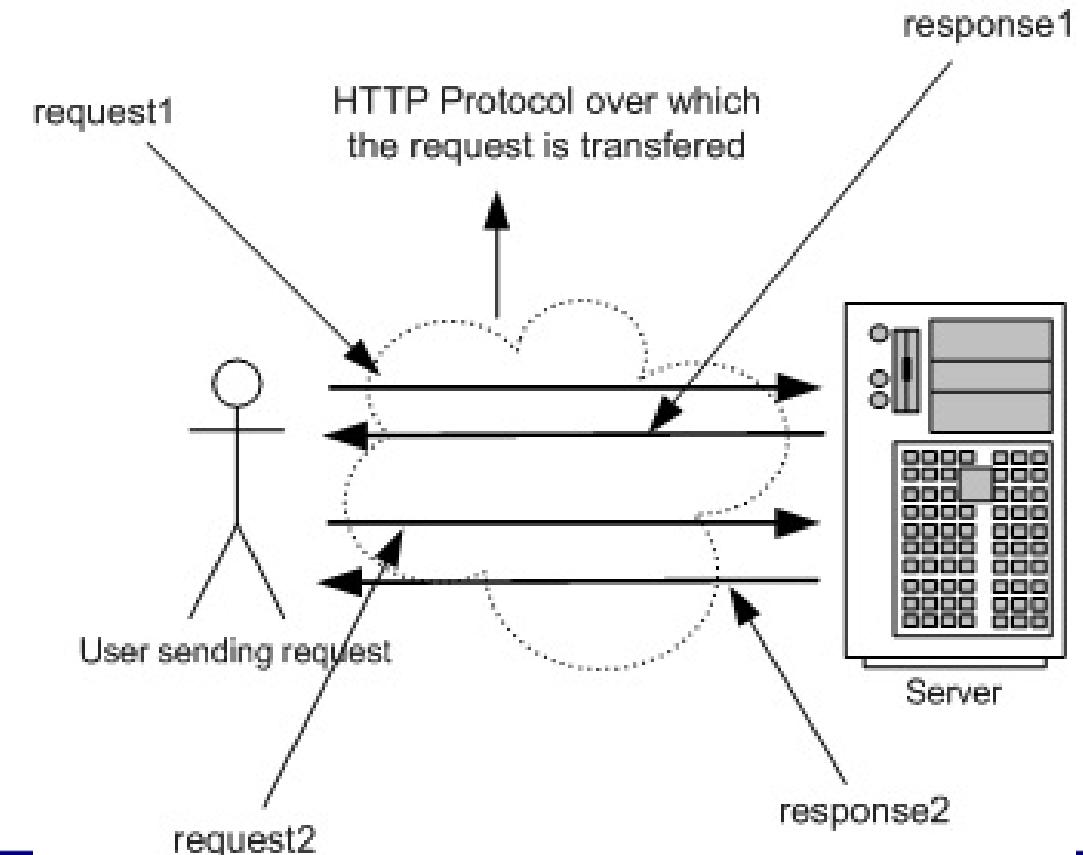
# Introduction to PHP Cookies and Sessions

- When the user utilizes a web application to accomplish task(s)/transaction(s), from the point the user **starts using** the application up to the point the user **leaves** the application, it can be seen as a **user session**.
- There can be some common data to be used by the application throughout the user session.
  - E.g. – User details (user name), logging details (authorization), items in the cart, etc...



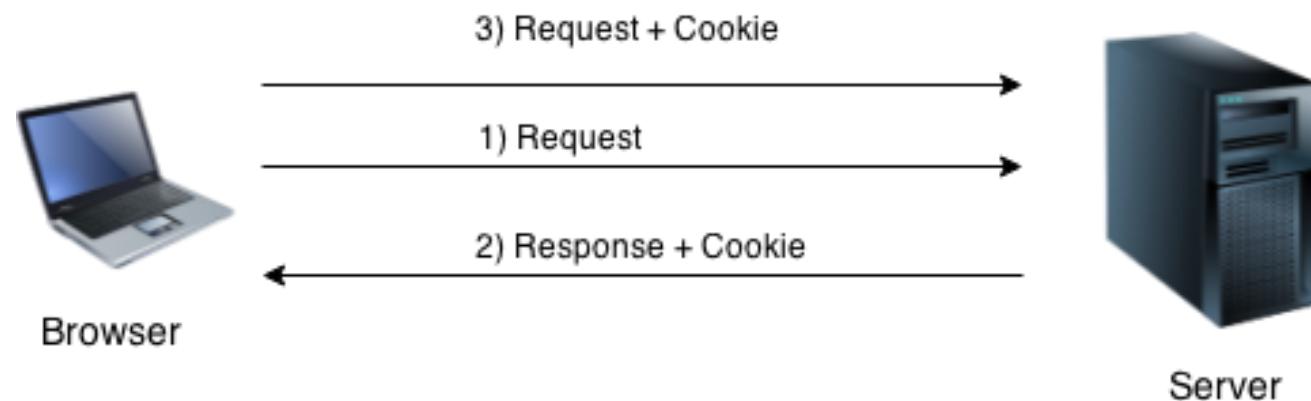
# Introduction to PHP Cookies and Sessions

- **HTTP is a stateless protocol**
  - It cannot maintain details/data between multiple requests
  - If need to share data between pages/requests, application level mechanism should be used
- PHP use techniques named **Cookies** and **Sessions**



# Cookies - Introduction

- Cookie is a small entry, which is saved in the user's device, by the server
  - Usually managed by the browser in a secured way
- Once a cookie is created/set, it will be sent back to the server with each request
- Cookies can also be processed by JS
  - Because it is saved in the client-side



# Cookies - Set a cookie

- A cookies is set as a **name:value** pair with some additional details

```
// setcookie(name, value, expire, path, domain, secure, httponly);
```

```
setcookie("Name", "SLIIT", time() + (86400 * 30), "/");
```

Time in seconds  
86400 = 1 day

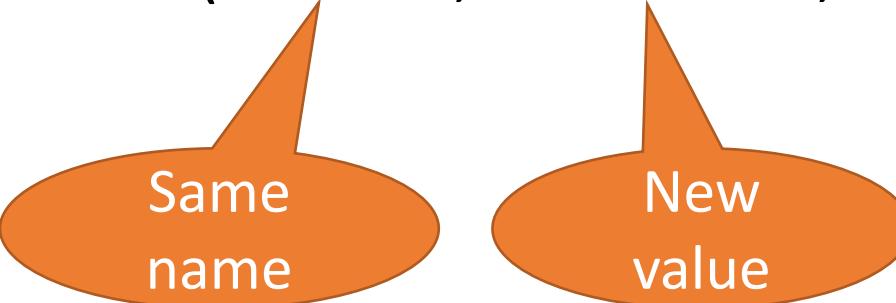
available for the entire  
website  
(or specify the dir)

# Cookies - Set a cookie

- You can see the cookie in the browser.
- Cookie will be expired after the set time
- You can remove the cookie(s) in the browser, using the browser settings

# Cookies - Modify a cookie

- To modify a cookie, you may set it again with the new value (and other data)
- `setcookie("Name", "SLIIT", time() + (86400 * 30), "/");`
- `setcookie("Name", "SLIIT FoC", time() + (86400 * 30), "/");`



Same  
name

New  
value

# Cookies - Delete a cookie

- To delete a cookie, modify it with a past expiration time

```
setcookie(" Name ", "", time() - 3600);
```



# Cookies- Use cookies

- Cookies are accessible using the `$_COOKIE` super global,
  - use the cookie name for the index

```
echo $_COOKIE["Name"]
```

# Cookies - Validate cookies

- You can check if the cookies are enabled for the application by checking the cookie count
  - `count($_COOKIE)`

```
if(count($_COOKIE) > 0) {
 echo "Cookies are enabled.>";
}

else {
 echo "Cookies are disabled.>";
}
```

# Cookies - Validate cookies

- You can check for a specific cookie using the `isset()`

```
if(isset($_COOKIE["Name"])) {
 echo $_COOKIE["Name"];
}

else {
 echo "No such cookie";
}
```

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($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!
";
 echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

<p>Note: You might have to reload the page to see
the value of the cookie.</p>

</body>
</html>
```

## Output?

Cookie named 'user' is not set!

**Note:** You might have to reload the page to see the value of the cookie.



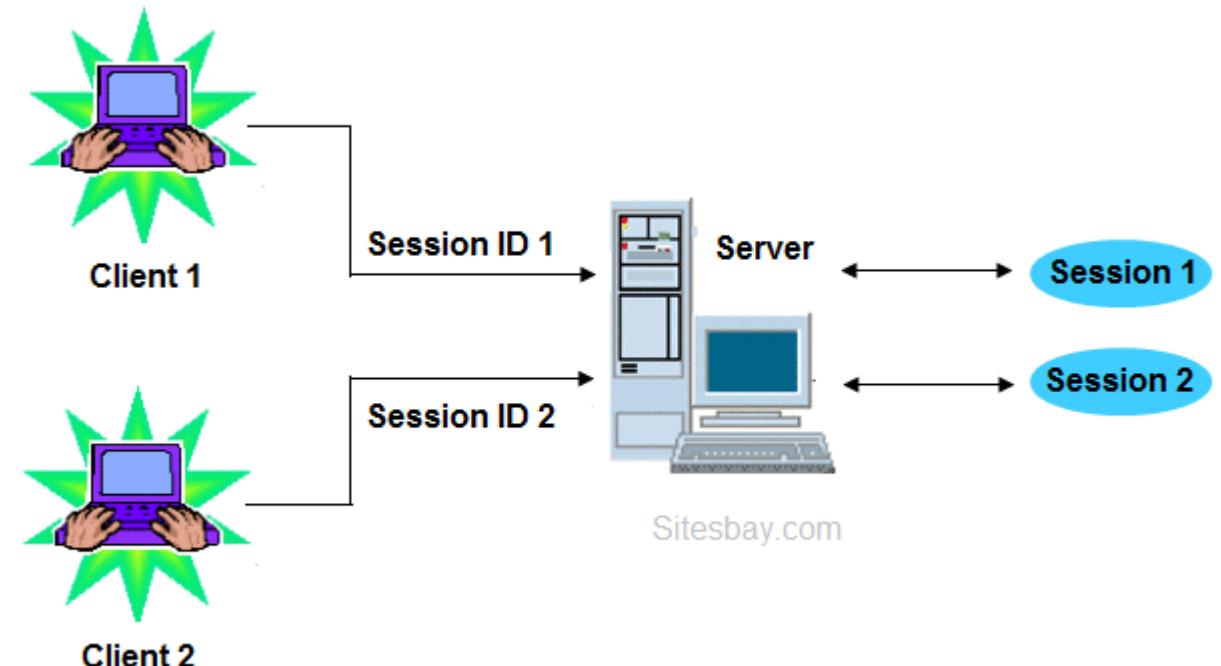
Cookie 'user' is set!

Value is: John Doe

**Note:** You might have to reload the page to see the value of the cookie.

# Sessions - Introduction

- Sessions are stored in the server
- PHP server can identify the user session and maintain the Session variables within a user session
- Unlike cookies, Sessions are discarded when the user leaves the application (closes the browser)



# Sessions - Introduction

- When the Session variables are used in an application, you have to start the session on each and every page, which use sessions

//This should be the first line of the page/file

**session\_start();**

# Sessions - Set/modify Session variables

- A Session variable is also a **name:value** pair
  - Use the `$_SESSION` super global variable
  - Name of the Session entry is used as the index

```
$_SESSION["Name"] = "SLIIT";
```

- To modify, you can simply assign the new value

```
$_SESSION["Name"] = "SLIIT FoC";
```

# Sessions - Set/modify Session variables

- A Session variable is also a **name:value** pair
  - Use the `$_SESSION` super global variable
  - Name of the Session entry is used as the index

```
$_SESSION["Name"] = "SLIIT";
```

- To modify, you can simply assign the new value

```
$_SESSION["Name"] = "SLIIT FoC";
```

# Sessions - Use Session variables

- Session variables can be used in the same way as a regular variable

```
echo $_SESSION["Name"];
```

# Sessions - End Session variables

- Session variables can be explicitly discarded by the application

```
// remove all session variables
```

```
session_unset();
```

```
// destroy the session
```

```
session_destroy();
```

E.g. :When the user logs off from the application

# Sessions - Validate Session variables

- `isset()` can be used to check if the Session variable is available

```
if(isset($_SESSION["Name"])) {
 echo $_SESSION["Name"];
}

else {
 echo "No such Session";
}
```

# Sessions - Validate Session variables

- empty() can be used to check if the Session variable contains a value

```
if(!empty($_SESSION[“Name”])) {
 echo $_SESSION[“Name”];
}

else {
 echo “Session has no value”;
}
```

```
<?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>
```

# Some use of the sessions

- The common uses of Sessions are
  - To maintain the user details (If the user is logged in or not)
  - To store the items in a shopping cart

# User log in – index.php

```
<html>
<head>
</head>
<body>
<h1>Log in</h1>
<form method="post" action="index.php">
 Username: <input type="text" name="txtName" />

 Password: <input type="password" name="txtPass" />

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

# User log in – index.php

- If the request has the `$_POST["txtName"]`, then it is the log in form submission

```
if(isset($_POST["txtName"]))
{
 //Validate the user
 //If a validuser, set a Session
}
```

# User log in – index.php

- User can be validated against the data in a DB
  - The sample code below validates the user against some static data just for demonstration

```
if($_POST["txtName"]=="asd" && $_POST["txtPass"]=="123")
{
 //Valid user, so set the Session
 $_SESSION["userName"] = $_POST["txtName"];
}
```

# User log in – index.php

- You may also check, if this is an already logged user.
  - We do not want to show a log in page to an already logged in user

```
if(isset($_SESSION["userName"]))
{
 //Redirect to another page
 header("Location:home.php");
}
```

# User log in – index.php

```
<?php //TOP OF THE PAGE
session_start();

if(isset($_POST["txtName"])){
 if($_POST["txtName"]=="asd" && $_POST["txtPass"]=="123") {
 $_SESSION["userUserName"] = $_POST["txtName"];
 }
}

if(isset($_SESSION["userUserName"])) {
 header("Location:home.php");
}
?>
```

# User log in – Other pages (home.php)

- We can identify if the user is an already logged in user or not by checking the Session.

```
<?php
session_start();

$userName = "";
if(isset($_SESSION["userName"])) { //Already logged in
 $userName = $_SESSION["userName"]; //Use the session value
}
else { // Not logged in
 header("Location:index.php"); //Redirect to the login page
}
?>
```

# User log in – Other pages (home.php)

- Other pages may have a log out feature

```
<html>
<head>
</head>
<body>
 <h1>Hello <?php echo $userName; ?></h1>
 <form method="post" action="logoff.php">
 <input name="logoff" type="submit" value="Log off"/>
 </form>
</body>
</html>
```

# User log off – logoff.php

- Check if the request is coming from a logoff feature
  - If coming from a log off feature, then end the Session and redirect to log in page

```
if(isset($_POST["logoff"]))
{
 session_destroy();
 header("Location:index.php");
}
```

# User log off – logoff.php

- If someone is trying to visit the page directly, we had better redirect to a proper page

```
else
{
 header("Location:home.php");
}
```

# User log off – logoff.php

```
<?php
session_start();

if(isset($_POST["logoff"])) {
 session_destroy();
 header("Location:index.php");
}
else {
 header("Location:home.php");
}
?>
```

# Summary

- Introduction to PHP Cookies and Sessions
- Cookies
- Sessions
- Some use of the sessions

# Lecture 11

# Search Engines and Social Media

IT1100 – Internet and Web Technologies

# Content

- ▶ Search Engines
- ▶ Social Media
- ▶ Accessing Social Media Data using PHP

# Search Engines

# What is a Search Engine?

A **Search Engine** is a computer programme that search documents for **specific keywords** and return a list of the documents where the keywords were found.

~ Webopedia ~

# Common Search Engines and Tools

- ▶ Desktop Search Tool
- ▶ Metasearch Engine
- ▶ Blog Search Engine
- ▶ Enterprise Search Tools

# **Web Search Engine**

A Web Search Engine is a tool that enables users to locate information on the World Wide Web.

# Search Engines

- ▶ Search engines help to search for content / information / resources in WWW
  - ▶ Web pages, text, files, images, audio, video, etc.
- ▶ The user input the search text (query – in key words or phrase) and get the search engine results pages (SERPs) as the output

# Search Engines

- ▶ Search engines maintain the details of the WWW, in order to produce the results faster.
- ▶ Web sites/applications maintain the details to be identified by the search engines.
- ▶ Meta data (using meta elements).

# **Seven Simple Changes to make Websites more visible to Search Engines**

1. Optimize every page of the website with rich content.
2. Write unique title tags and meta descriptions for every page.
3. Weed out 404 errors.
4. Make the page/site content easily shareable.
5. Optimize images.
6. Make the Website load faster.
7. Include structured markup.

<https://www.entrepreneur.com/article/273801>

# Search Engines

- ▶ A search engine maintains the following processes in near real time.
  - ▶ Web Crawling
  - ▶ Indexing
  - ▶ Searching

# Web Crawling

- ▶ Search engines use an application called web crawler (*also called spider or bot*) to systematically crawl/browse through the content in the WWW.
- ▶ The web sites are optimized for the crawler to access, identify, and understand the content easily.
- ▶ Web site/application owners can submit their domain details to search engines to ensure the indexing.

# Web Crawling

- ▶ Web crawler uses policies to have optimal results, yet not damaging the web site
  - ▶ Selection Policy
  - ▶ Re-visit Policy
  - ▶ Politeness Policy
  - ▶ Parallelization Policy

# **Indexing**

- ▶ The details/data gathered by crawling are stored, for fast and accurate retrieval.
- ▶ The result can be seen as an index of web resources.

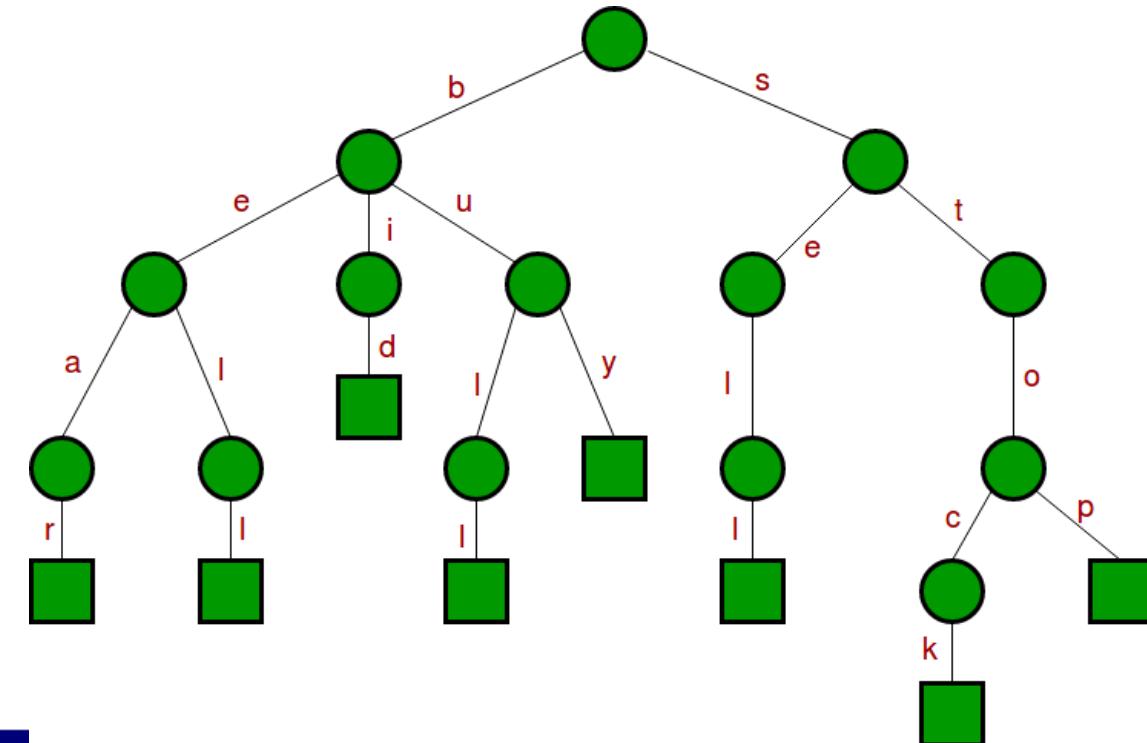
# Indexing

- ▶ Special index data structures are used to improve the processing productivity.

- ▶ Suffix trees
- ▶ Inverted index
- ▶ Document-term matrix

	I	like	hate	databases
D1	1	1	0	1
D2	1	0	1	1

{bear, bell, bid, bull, buy, sell, stock, stop}



# Searching

- ▶ User can specify keywords/phrases and search for content/resources.
- ▶ User's input is called a **web search query**.
- ▶ Search engines use query languages to process the query and retrieve matching content from the indexes.
- ▶ Then the matching content are shown to the user with hyperlinks to reach the original sources

# Social Media

# What is Media?

- ▶ Media is the plural of medium.
- ▶ Any form of communicating.
  - ▶ Newspapers
  - ▶ Magazines
  - ▶ Radio
  - ▶ Television
  - ▶ Internet (*considered as a group*)

# What is Multimedia?

Multimedia means using a combination of moving and still pictures, sound, music, and words, especially in computers or entertainment.

[Cambridge dictionary]

# What is Social Media?

websites and computer programs that allow people to communicate and share information on the internet using a computer or mobile phone.

[Cambridge dictionary]

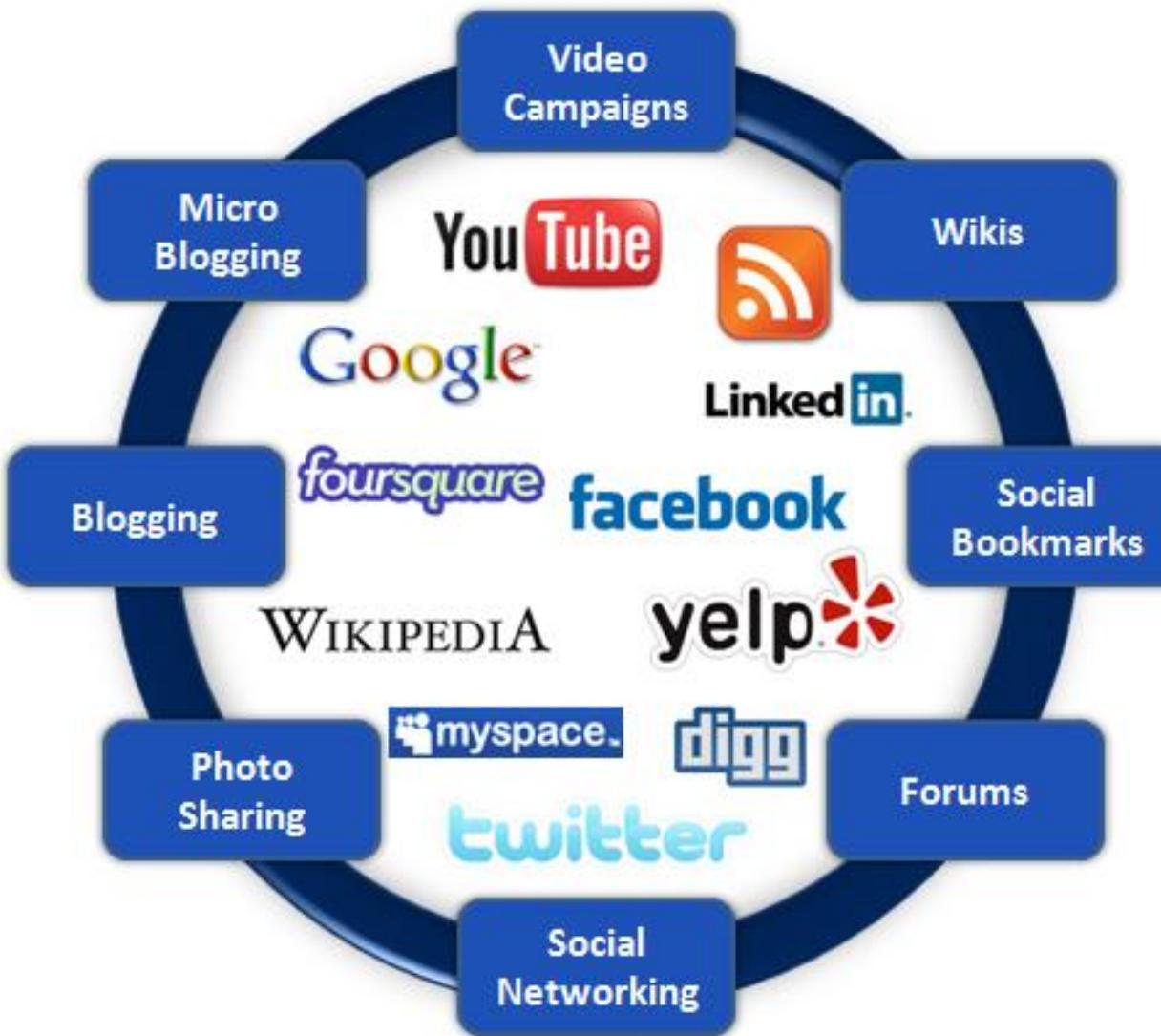
# What is Social Media?

- ▶ Web sites/applications that allow people to communicate and share information, ideas, career interests, and other forms of expressions on the internet using a computer or mobile phone.
- ▶ Usually the information is shared within social groups.

# Uses of Social Media

- ▶ Communication.
- ▶ Collaboration.
- ▶ Opinions & Reviews.
- ▶ Brand Monitoring.
- ▶ Entertainment.
- ▶ Media Sharing.
- ▶ Paid Advertising.

# Different types of Social Media



## Types of Social Media

# Social Networking

- ▶ Find friends with similar interest, group activities, share resources and events.



- ▶ Chat – Communication between individuals or groups, publishing announcements, resource sharing.



# Types of Social Media

## Blogging

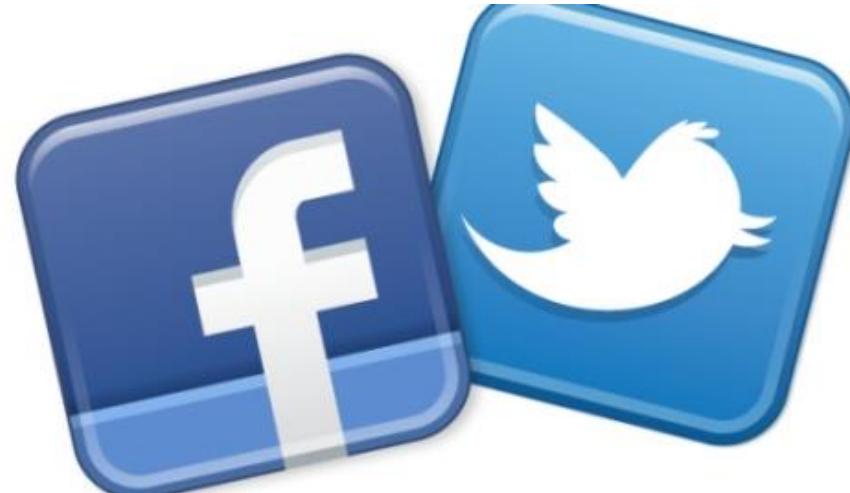
- ▶ Content writing (*articles, tutorials, technical reviews, stories, etc.*) and publishing, commenting, and reviewing.



# Types of Social Media

## Micro-blogging

- ▶ Status updates within groups, friends



## Types of Social Media Photo Sharing

- ▶ Share/publish photos/images: personals, professional, hobbies, etc.



## Types of Social Media Video Sharing

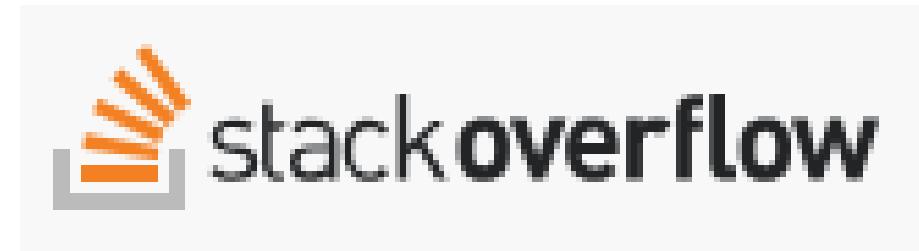
- ▶ Share/publish videos: Movies, songs, documentaries, technical, etc.



# Types of Social Media Forums

- ▶ Ask questions, share experience/knowledge, learn.

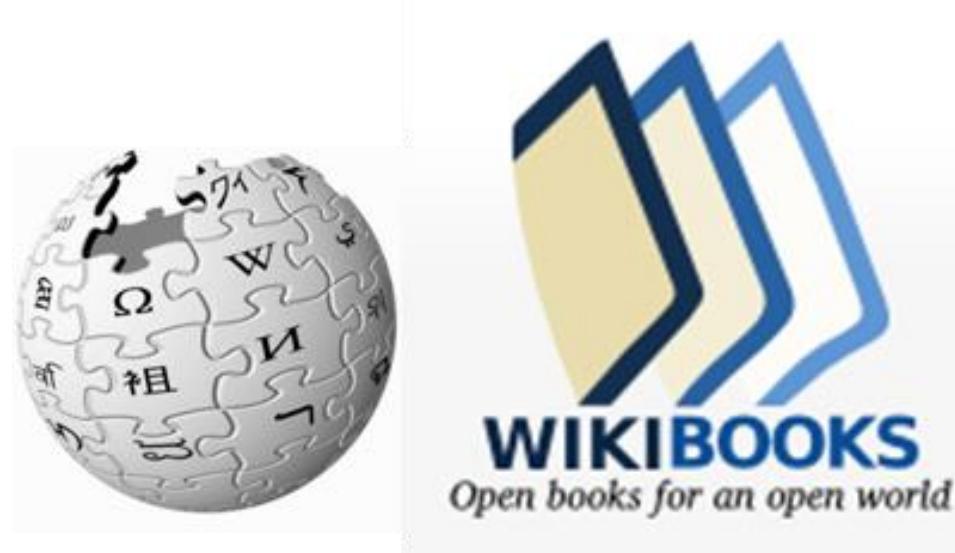
Quora



# Types of Social Media

## Wikis

- ▶ Content is built by the community as a group.



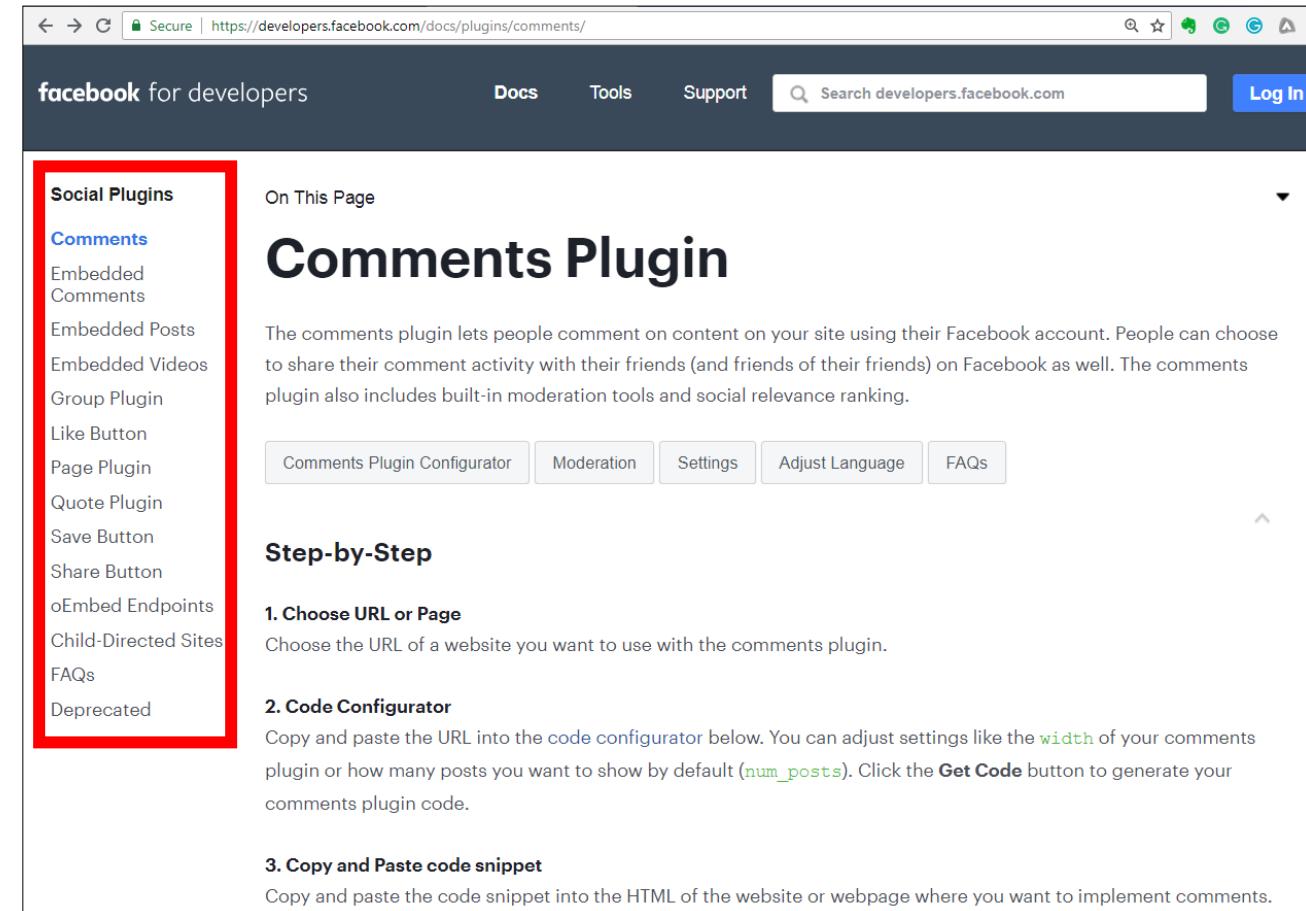
# Social Media Integration

- ▶ Popular social media sites/applications provides various widgets, which can be integrated into other systems.
  - ▶ *Google logging, Facebook comments, YouTube video*
- ▶ Usually the social media site/application provides the integration details.

# Social Media Integration Available Social Plugins

## ► Facebook comments plugin

► <https://developers.facebook.com/docs/plugins/comments/>



The screenshot shows a web browser displaying the Facebook Developers documentation for the Comments Plugin. The URL in the address bar is <https://developers.facebook.com/docs/plugins/comments/>. The page title is "Comments Plugin". On the left, there is a sidebar menu titled "Social Plugins" with a red box highlighting the "Comments" option. The main content area contains a brief description of the Comments Plugin, followed by three numbered steps: 1. Choose URL or Page, 2. Code Configurator, and 3. Copy and Paste code snippet.

Secure | https://developers.facebook.com/docs/plugins/comments/

facebook for developers Docs Tools Support Search developers.facebook.com Log In

Social Plugins

Comments

- Embedded Comments
- Embedded Posts
- Embedded Videos
- Group Plugin
- Like Button
- Page Plugin
- Quote Plugin
- Save Button
- Share Button
- oEmbed Endpoints
- Child-Directed Sites
- FAQs
- Deprecated

On This Page

## Comments Plugin

The comments plugin lets people comment on content on your site using their Facebook account. People can choose to share their comment activity with their friends (and friends of their friends) on Facebook as well. The comments plugin also includes built-in moderation tools and social relevance ranking.

Comments Plugin Configurator Moderation Settings Adjust Language FAQs

### Step-by-Step

- 1. Choose URL or Page**  
Choose the URL of a website you want to use with the comments plugin.
- 2. Code Configurator**  
Copy and paste the URL into the code configurator below. You can adjust settings like the `width` of your comments plugin or how many posts you want to show by default (`num_posts`). Click the **Get Code** button to generate your comments plugin code.
- 3. Copy and Paste code snippet**  
Copy and paste the code snippet into the HTML of the website or webpage where you want to implement comments.

# Social Media Integration Google Sign-In

## ► Google Sign-In

► <https://developers.google.com/id/entity/sign-in/web/sign-in>

Integrating Google Sign-In into your web app ★★★★★

Google Sign-In manages the OAuth 2.0 flow and token lifecycle, simplifying your integration with Google APIs. A user always has the option to [revoke access](#) to an application at any time.

This document describes how to complete a basic Google Sign-In integration.

### Before you begin

Before you can integrate Google Sign-In into your website, you must create a client ID, which you need to call the sign-in API.

To create a Google API Console project and client ID, click the following button:

[CONFIGURE A PROJECT](#)

When you configure the project, select the **Web browser** client type and specify the origin URI of your app.

After configuration is complete, take note of the client ID that was created. You will need the client ID to complete the next steps. (A client secret is also created, but you need it only for server-side operations.)

# Accessing Social Media Data Using PHP

# Accessing Social Media Data Using PHP

## ► Facebook SDK v5 for PHP

- ▶ Enable PHP developers to easily integrate Facebook login and make requests to the Graph API.
- ▶ Makes it easy to upload photos and videos and send batch requests to the Graph API among other things.
- ▶ Whether you're developing a website with Facebook login, creating a Facebook Canvas app or Page tab, the Facebook SDK for PHP does all the heavy lifting for you making it as easy as possible to deeply integrate into the Facebook platform.

# References

<https://developers.facebook.com/docs/reference/php/>

The screenshot shows a web browser displaying the Facebook SDK v5 for PHP documentation. The URL in the address bar is <https://developers.facebook.com/docs/reference/php/>. The page title is "Facebook SDK v5 for PHP". On the left, there's a sidebar with links for "Web SDKs" (JavaScript SDK, PHP SDK), "Getting Started", and "Reference". The main content area starts with a brief introduction about the PHP SDK, mentioning its integration with the Graph API and its extensibility. It then lists examples under "Examples", which include "Authentication & Signed Requests", "User profile", and "File Uploads". Each example has a corresponding list of tasks or operations. The browser interface at the bottom includes the Windows taskbar with various pinned icons like File Explorer, Google Chrome, and Microsoft Edge.

# References

<https://developers.facebook.com/>

A screenshot of the Facebook for Developers website. The top navigation bar includes links for Products, Docs, More, Log In, and a search icon. A large banner on the right features the text "ONNX" and "Learn about AI and how we are making it more open and interoperable." with a "Watch Now" button. Below the banner is a small set of four dots. The main content area on the left displays a street scene with several cars highlighted by translucent green and blue bounding boxes, illustrating AI object detection. At the bottom, there are three sections: "Winners of the Developer Circles Community Challenge" (with a "Learn More" link), "Manage App Roles and Get Notifications in Facebook Business Manager" (with a "Learn More" link), and "Regional Winners for the Developer Circles Community Challenge Announced" (with a "Learn More" link). The Windows taskbar at the bottom shows various pinned icons.

# References

<https://developers.google.com/>

The screenshot shows the Google Developers website with the URL https://developers.google.com/ in the address bar. The page features a navigation bar with links for Home, Product Index, Events, Developer Programs, and Blog. The main heading is "Build anything with Google". Below it is a search bar with the placeholder "Search products and documentation". A "Events" section is displayed, featuring two event cards. The first card is for the "Android Dev Summit 2018" held from November 7-8 in Mountain View, USA. It shows a group of people in an audience and includes a "LEARN MORE" button. The second card is for the "Chrome Dev Summit 2018" held from November 12-13 in San Francisco, USA. It shows a group of people seated on chairs and includes a "LEARN MORE" button. The bottom of the screen shows the Windows taskbar with various pinned icons.



# Summary

- ▶ Search Engines
- ▶ Social Media
- ▶ Accessing Social Media Data using PHP