

Web Programming II

(Server Side Scripting)

Copyright

This course has been developed as part of the collaborative advanced ICT course development project of the Commonwealth of Learning (COL). COL is an intergovernmental organization created by Commonwealth Heads of Government to promote the development and sharing of open learning and distance education knowledge, resources and technologies.

The Open University of Tanzania (OUT) is a fully fledged, autonomous and accredited public University. It offers its certificate, diploma, degree and postgraduate courses through the open and distance learning system which includes various means of communication such as face-to-face, broadcasting, telecasting, correspondence, seminars, e-learning as well as a blended mode. The OUT's academic programmes are quality-assured and ascentrally regulated by the Tanzania Commission for Universities (TCU).



© 2016 by the Commonwealth of Learning and The Open University of Tanzania. Except where otherwise noted, *Web Programming II (Server Side Scripting)* is made available under Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) License: <https://creativecommons.org/licenses/by-sa/4.0/legalcode>.

For the avoidance of doubt, by applying this licence the Commonwealth of Learning does not waive any privileges or immunities from claims that it may be entitled to assert, nor does the Commonwealth of Learning submit itself to the jurisdiction, courts, legal processes or laws of any jurisdiction. The ideas and opinions expressed in this publication are those of the author/s; they are not necessarily those of *Commonwealth of Learning* and do not commit the organisation.



The Open University of Tanzania
P. O. Box 23409,
Kinondoni,
Dar Es Salaam,
Tanzania
Phone: +255 22 2668992
Fax: +255 22 2668756
Email: hod.ict@out.ac.tz
Website: www.out.ac.tz



Commonwealth of Learning
4710 Kingsway, Suite 2500,
Burnaby V5H 4M2,
British Columbia,
Canada
Phone: +1 604 775 8200
Fax: +1 604 775 8210
Email: info@col.org

Acknowledgements

The Open University of Tanzania (OUT), Faculty of Science, Technology and Environmental Studies, (FSTES) and the ICT Department wish to thank those below for their contribution to the production of this course material and video lectures:

Authors:

Mr. Mathias Ombeni (Assistant Lecturer, ICT Department, OUT)

Mr. Emmanuel Thomas (Technologist, IEMT, OUT)

Copy Editor:

Mr. Justin Kimaro (Principal Editor, IEMT, OUT)

Reviewers:

Lilian C. Mutalemwa (Assistant Lecturer, ICT Department, OUT)

Regina Monyemangene (Instructional Designer, IEMT, OUT)

Video Production:

Mr. Othman Mwinchom (Multimedia studio, IEMT, OUT)

Ms. Grace W. Mbwete (Assistant Lecturer, ICT Department, OUT)

Mr. Mathis Ombeni (Assistant Lecturer, ICT Department, OUT)

This course has been developed with the support of the Commonwealth of Learning, Canada.

Contents

About this COURSE MATERIAL	1
How this Course Material is structured	1
Course overview	3
Welcome to: Web Programming II.....	3
Web Programming II—is this course for you?.....	4
Course outcomes.....	4
Timeframe.....	5
Need help?	5
Assessments.....	6
Getting around this COURSE MATERIAL	7
Margin icons	7
Unit 1	9
Client Server Architecture and Web Protocols.....	9
Introduction	9
Unit outcomes.....	9
1.1 Client/Server Evolution.....	10
1.2 Two Sides of Web Programming	11
1.3 Advantages of Client/Server Architecture	12
1.4 Disadvantages of Client/Server Model.....	12
1.5 Communication between Browser and Web Sever.....	13
Activity	16
Unit summary	16
Review Questions	17
Reference and Further Reading	17
Attribution.....	18
Unit 2	19
Introduction to Sever Side Scripting and Technologies	19
Introduction	19
Unit outcomes.....	19
2.1 How to Choose Web Programming Languages	20
2.2 List of Popular Programming Languages.....	21
2.2.1 Java/Servlet.....	21
2.2.3 Perl	22
2.2.4 PHP	23
2.2.5 C Language	24

2.2.6 ASP	24
2.2.7 ColdFusion	25
2.2.8 AJAX	25
2.2.9 jQuery	26
2.2.10 Common Gateway Interface (CGI).....	26
Activity	28
Unit summary	29
Review Questions	29
Reference and Further Reading	30
Attribution.....	31
Unit 3	32
PHP Scripting Language.....	32
Introduction	32
Unit outcomes.....	32
3.1 About PHP.....	33
3.2 Variable and Data Types in PHP.....	34
3.3 Operators	37
3.4 Input Output	38
3.5 Control Structures.....	40
3.6 Arrays	41
3.7 Functions	44
3.8 Strings.....	46
3.9 Regular Expressions	48
Activity	56
Unit summary	57
Review Questions	58
Reference and Further Reading	58
Attribution.....	59
Unit 4	60
Interacting with Database	60
Introduction	60
Unit outcomes.....	60
4.1 Understand MySQL	61
4.2 Simple SQL Retrieval	62
4.3 PHP Database Functions	63
Activity	66
Unit summary	68
Review questions	68
Reference and Further Reading	69
Attribution.....	69
Unit 5	70
Sessions and Cookies in PHP	70
Introduction	70

Unit outcomes.....	70
5.1 Understand Sessions.....	71
5.2 Understand Cookies.....	72
5.3 Uses of Cookies.....	72
5.4 How to Create a Cookie in PHP.....	73
5.5 How to Delete Cookies.....	75
5.6 Retrieving cookie data.....	76
5.7 Where are cookies used?.....	76
Activity.....	78
Unit summary.....	78
Review Questions.....	79
Reference and Further Reading.....	79
Attribution.....	79
Unit 6	81
Introduction to AJAX.....	81
Introduction.....	81
Unit outcomes.....	81
6.1 AJAX Basics.....	82
6.1.1 Asynchronous.....	82
6.1.2 JavaScript.....	82
6.1.3 XML.....	82
6.2 How Ajax works.....	83
6.3 Importance of Ajax.....	83
6.4 XMLHttpRequest Object.....	84
6.5 How to Send Request to a Sever.....	85
6.6 Request Methods.....	87
7.6 .5 AJAX Security Concerns.....	95
Activity.....	97
Unit summary.....	98
Review Questions.....	98
Reference and Further Reading.....	98
Attribution.....	99
Unit 7	100
Introduction to Perl.....	100
Introduction.....	100
Unit outcomes.....	100
Terminologies.....	101
7.1 Perl Background.....	101
7.1.1 History of Perl.....	101
7.1.2 Perl Development Life Cycle.....	102
7.1.4 Comments.....	103
7.2 How Perl Works.....	105
7.3 Scalars and Scalar variables.....	106
7.4 Perl Functions.....	109

7.4.1 Print.....	110
7.4.2 Length	110
7.4.3uc / lc.....	111
7.4.4 Reverse.....	111
7.4.5 Substr	111
Activity	112
Unit summary	114
Review questions	114
Reference and Further Reading	114
Attribution.....	115
Unit 8	116
Introduction to jQuery	116
Introduction	116
Unit outcomes.....	116
Terminologies.....	116
8.1 Understanding jQuery	117
8.1.1 What is JQuery.....	117
8.1.2 What Can jQuery Do	117
8.1.3 jQuery - Event Handling.....	118
8.1.4 Downloading jQuery.....	118
8.2 jQuery Selector.....	119
8.2.2 The Element Selector.....	120
8.2.3 The #Id Selector.....	120
8.2.3 The Class Selector.....	121
8.3 jQuery Events	121
Activity	122
Unit summary	124
Review Questions	125
Reference and Further reading.....	125
Attribution.....	125
The following material was adapted from the material:	125
Unit 9	127
Introduction to Common Gateway Interface (CGI).....	127
Introduction	127
Unit outcomes.....	127
9.1 Understanding Common Gateway Interface (CGI).....	128
9.2 How CGI Works.....	129
9.3 CGI Deployment	133

Activity	135
Unit summary	136
Review questions	136
Reference and Further reading	136
Attribution.....	137
Unit 10	138
<hr/>	
Web Application Security.....	138
Introduction	138
Unit outcomes.....	138
10.1 Understanding Web Application Security.....	139
10.1.1 The primary web security controls	139
10.1.2 Common words used in computer security.....	140
10.2 Main Types of Website Attacks	141
10.2.1 List of types of website's and web system's attacks.....	141
10.2.2 Common mistakes that introduces security holes in websites and web systems	141
10.2.3 How Hackers Detect existence of security holes in Websites or Web systems	141
10.3 Detection and Solution of the Attacks.....	142
10.3.1 Categories of attacks	142
10.3.2 Preventing websites and web applications from being hacked.....	142
10.3.3 Solution to website's and web system's attacks.	142
Activity	144
Unit summary	145
Review questions	145
Reference and Further reading	146
Attribution.....	146



About this COURSE MATERIAL

Web Programming II (server-side scripting) has been produced by The Open University of Tanzania in collaborations with Commonwealth of Learning. This COURSE MATERIAL is structured as outlined below:

How this Course Material is structured

The course overview

The course overview gives you a general basic of server side programming language. Information contained in the course overview will help you determine:

- If the course is suitable for you.
- What you will already need to know.
- What you can expect from the course.
- How much time you will need to invest to complete the course.

The overview also provides guidance on:

- Study skills.
- Where to get help.
- Course assessments.
- Activity icons.
- Units.

We strongly recommend that you read the overview *carefully* before starting your study.

The course content

The course is broken down into units. Each unit comprises:

- An introduction to the unit content.
- Unit outcomes.
- New terminology.
- Core content of the unit with a variety of learning activities.
- A unit summary.
- Review questions
- Reference and Further Reading.



- Attribution.
- There are links to video lectures spread out in this material.

Resources

For those interested in learning more on this course, we provide you with website links in the notes and a list of additional resources at the end of each unit of this course materials; these may be books, articles or websites links. There are links to video lectures for each unit.

Your comments

After completing Web Programming II we would appreciate it if you would take a few moments to give us your feedback on any aspect of this course. Your feedback might include comments on:

- Course content and structure.
- Course reading materials and resources.
- Course Activities.
- Video lectures.
- Course review questions.
- Course duration.
- Course support (assigned tutors, technical help, etc.)

Your constructive feedback will help us to improve and enhance this course.



Course overview

Welcome to: Web Programming II(Server Side Scripting)

Nowadays, many applications work really well as a web application. Web programming is the practice of writing applications that run on a web server and can be used by many different people. Web programming allows you to turn a simple, static HTML page into a dynamic masterpiece. It allows others to interact with your web site and use the application on any computer with Internet access. It is often easier than programming applications that will run directly on the computer. It allows you to make or edit anything dynamic on your website, such as a forum, a guestbook, or even a form submission. This course will help you understand, what web programming is and why you might want to do it.

Course Overview Video

<https://tinyurl.com/lstrfqfb>





Web Programming II—is this course for you?

This course is intended for people who have basic web and internet skills.

The course aims are:

- To enable learners to understand the purpose of Client/Server Architecture.
- To enable learners use to different Server-Side scripting Languages.
- To enable learners develop database driven web application using different web programming languages and technologies.
- To enable learners to understand the types of web sites and web system's attacks and how to deal with them.

Course outcomes



Upon completion of Web Programming II (Server-side scripting), learners should be able to:

- Describe the purpose of Client/Server Architecture
- Explain Internet Protocols.
- Demonstrate how web works.
- Describe importance of Web Programming languages
- Develop database-driven application using PHP and MySQL
- Write code to create session and cookies in PHP application
- Develop application using AJAX to communicate and exchange data to and from server and database.
- Describe different types of websites and web system's attacks.
- Use various tools and techniques to implement web security



Timeframe



This is a 6 months course.

This course requires timeframe depends on individual institution's mode of delivery.

A minimum standard of delivery should be 6 weeks of formal lectures, 12 weeks of supervised laboratory tutorials and 6 weeks of unsupervised directed learning.

Self-study time is 10 hours per-week.

Need help?



Help

This course is offered at The Open University of Tanzania, ICT department

If you need help regarding this course, please contact:



Head of Department, ICT department
The Open University of Tanzania
P. O. Box 23409,
Kinondoni,
Dar Es Salaam,
Tanzania.
Phone: +255 22 2668992
Fax: +255 22 2668756
Email: hod.ict@out.ac.tz
Website: www.out.ac.tz



Assessments



There are activities and review questions in each unit of this course.

Activities are assessed in three modalities:

- Peer – review
- Self – assessment
- Instructor – marked assessment

NB: Review questions are for self – assessment















Getting around this COURSE MATERIAL

Margin icons

While working through this COURSE MATERIAL you will notice the frequent use of margin icons. These icons serve to “signpost” a particular piece of text, a new task or change in activity; they have been included to help you to find your way around this COURSE MATERIAL.

A complete icon set is shown below. We suggest that you familiarize yourself with the icons and their meaning before starting your study.

			
Activity	Assessment	Assignment	Case study
			
Discussion	Group activity	Help	Note it!
			
Outcomes	Reading	Reflection	Study skills
			
Summary	Terminology	Time	Tip



Unit 1

Client Server Architecture and Web Protocols

Introduction

This unit begins with an overview of Client/Server Evolution. Students will learn all basics about Internet Protocols, focusing on the Hyper Text Transfer Protocol (HTTP), and the different methods by which certain servers are able to perform requests over the Internet.

Unit outcomes

Upon successful completion of this unit, you will be able to:



- Explain how Client-Server systems work.
- Differentiate between two-tier and three-tier architectures.
- Describe multi-tier architectures.
- Differentiate different Internet Protocols
- Explain how Hypertext Transfer Protocol works

Terminologies



HTTP:	Protocol for transmitting linked documents in the world wide world
Client:	Device that get services from the server
Server:	Provides services for other programs or devices
Web Browser:	Used to access information provided by web servers



User Agent:	Refers to both end points of a communications session.
Network Port:	Number that identifies one side of a connection between two computers
URL:	A reference (an address) to a resource on the Internet

1.1 Client/Server Evolution

A long time ago, client-server computing was just using mainframes and connecting to dumb terminals. Through the years, personal computers started to evolve and replaced these terminals but the processing is still process on the mainframes. With the improvement in computer technology, the processing demands started to split between personal computers and mainframes.

The term client-server refers to a software architecture model consisting of two parts, client systems and server systems. This two components can interact and form a network that connect multiple users. Using this technology, PCs are able to communicate with each other on a network. These networks were based on file sharing architecture, where the PC downloads files from corresponding file server and the application is running locally using the data received. However, the shared usage and the volume of data to be transferred must be low to run the system well. As the networks grew, the limitations of file sharing architectures become the obstacles in the client-server system.

This problem is solved by replacing the file server with a database server. Instead of transmitting and saving the file to the client, database server executes request for data and return the result sets to the client. In the results, this architecture decreases the network traffic, allowing multiple users to update data at the same time.

Typically either Structured Query Language (SQL) or Remote Procedure Calls (RPCs) are used to communicate between the client and server.



There are several types of client-server architecture. One of the architecture is the Two Tier Architecture, where a client is directly connected to a server. This architecture has a good application development speed and work well in homogeneous environments when the user population work is small. The problem exists in this architecture is the distribution of application logic and processing in this model. If the application logic is distributed to dozens of client systems, the application maintenance will be very difficult. To overcome the limitations of the Two-Tier Architecture, Three Tier Architecture is introduced. By introducing the middle tier, clients connect only to the application server instead of connect directly to the data server. By this way, the load of maintaining the connection is removed. The database server is able to manage the storage and retrieve the data well. Thus, the application logic and processing can be handled in any application systematically. To enhance the Three Tier Architecture, it can be extended to N-tiers when the middle tier provides connections to various types of services, integrating and coupling them to the client, and to each other. For example, web server is added to Three Tier Architecture to become Four Tier Architecture where the web server handles the connection between application server and the client.

Client side languages are languages that are sent to the client, and then processed (interpreted and displayed). JavaScript is a client side language.

Server side languages are languages that are processed on the server before the result is then sent to the client and displayed.

1.2 Two Sides of Web Programming

Any Web communication has two basic "sides" of the conversation: You have the client, which is typically the Web browser, and which initiates the communication, and you have the server, a computer that is configured to wait for and serve any incoming communication requests.

Web site development often involves writing programs telling computers what to do. But there are two very different types of Web



programming, separated by which of the two sides is to execute the program: It may be that we want the server to execute the program, or it may be that we want the Web browser to execute the program. Every Web program, then, can be called either server-side or client-side.

Both categories of programming are important in the modern Web, because each offers a distinct set of advantages and thus each is appropriate at different times. And in today's Web, they are completely separate, since they involve very different programming languages. Let us first look at their relative advantages, which dictate the instances where each applies.

1.3 Advantages of Client/Server Architecture

The client/server model is particularly recommended for networks requiring a high degree of reliability, the main advantages being:

- **Centralized resources:** given that the server is the centre of the network, it can manage resources that are common to all users, for example a central database would be used to avoid problems caused by redundant and inconsistent data.
- **Improved security:** as the number of entry points giving access to data is not so important
- **Server level administration:** as clients do not play a major role in this model, they require less administration
- **Scalable network:** Using this architecture it is possible to remove or add clients without affecting the operation of the network and without the need for major modification.

1.4 Disadvantages of Client/Server Model

Client/Server architecture also has the following drawbacks: increased cost: due to the technical complexity of the server a weak link: the server is the only weak link in the client/server network, given that the entire network is built around it! Fortunately, the server is highly fault tolerant (primarily thanks to the RAID system) Client/Server system operation.

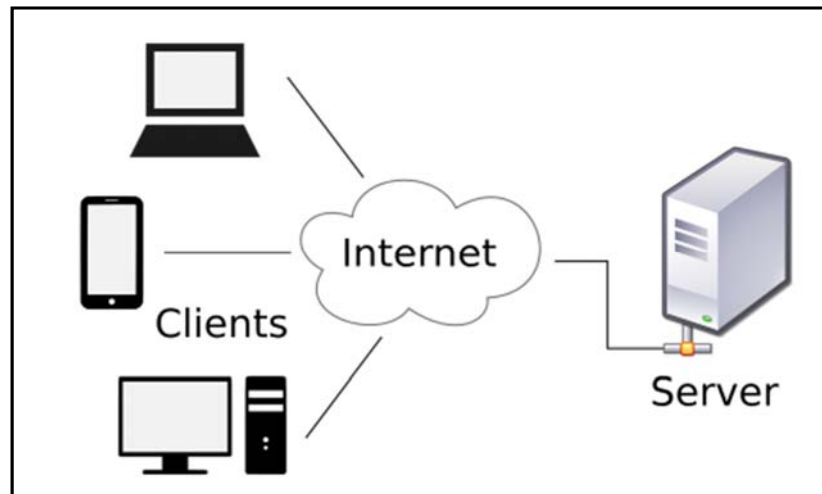
A client/server system operates as outlined in the following diagram:



1.5 Communication between Browser and Web Sever

When web browsers communicate with a web server, a number of important things happen during the conversation.

The browser and the server “talk” to each other using a relatively simple protocol called HTTP, which stands for the Hyper Text Transfer Protocol.



Source: <https://en.wikipedia.org/wiki/File:Client-server-model.svg>

HTTP can be broken into two components where we get a little more detail:

Hyper Text:

The idea of hyper text is that it links you from one page to another (we used to call the links (hyperlinks) but that has been shortened to just links. However, the concept of Hyper Text is central to this course in that the specific text that is transferred follows a standard called the Hyper Text Markup Language (HTML).

Transfer Protocol

When two programs want to talk to each other both programs have to “speak the same language.” A protocol is a very specific description of a series of messages that two programs use to communicate.



The web server sends the HTML to the browser over the internet and the browser renders the HTML into the nicely formatted page that you see on your screen.

Hypertext Transfer Protocol

HTTP is an application layer protocol provided by IP. The messages and methods that allow communication between a Web Server and a Web Client are introduced. Read this webpage and make sure you understand Figure 1, which illustrates the HTTP Request Message Format.

i) Hypertext Transfer Protocol

Often abbreviated as HTTP, this is a communications protocol. It is used to send and receive webpages and files on the Internet. It was developed by Tim Berners-Lee and is now coordinated by the W3C. HTTP version 1.1 is the most common used version today. HTTP works by using a user agent to connect to a server. The user agent could be a web browser or spider. The server must be located using a URL or URI. This always contains `http://` at the start. It normally connects to port 80 on a computer.

A more secure version of HTTP is called HTTPS: This contains `https://` at the beginning of the URL. It encrypts all the information that is sent and received. This can stop malicious users such as hackers from stealing the information. HTTPS is often used on payment websites. HTTPS uses port 443 for communication instead of port 80.

ii) Request Message

The request message contains the following:

- Request line, such as `GET /images/logo.gif HTTP/1.1`, which requests the file `logo.gif` from the `/images` directory
- Headers, such as `Accept-Language: en`
- An empty line
- An optional message body

The request line and headers must all end with two characters: a carriage return followed by a line feed, often written `<CR><LF>`. The empty line



must consist of only <CR><LF> and no other whitespace. In the HTTP/1.1 protocol, all headers except Host are optional.

A request line containing only the path name is accepted by servers to maintain compatibility with HTTP clients before the HTTP/1.0 standard. Even this site has a HTTP at it's beginning, which is: HTTP (Hyper Transfer Protocol) is a communication protocol and it is used to send and receive webpages and other data files on the internet. A more secure version of HTTP is called HTTPS (Hyper Transfer Protocol Secure). It encrypts all the information sent and received. This can stop hackers from stealing the information; HTTPS is often used on payment websites.

Video Lecture

<https://tinyurl.com/kxa929t>




<https://tinyurl.com/kxa929t>





Activity

	<p>Activity 1.0 Finding function and tools in client- server application architecture</p> <p>Aim: Describe how 3- tier application architecture works</p> <p>Motivation: To become conversant with functions and tools involved in client-server architectures</p> <p>Resources: Unit 1 lecture notes</p> <p>What to do:</p> <p>Draw the 3-tier application architecture for database driven website. Discuss the functions of each layer, and define the necessary tools involved.</p> <p>Duration: Expect to spend about 1 hour on this activity</p> <p>Feedback: This is a peer-reviewed activity where the learner should share their findings with other learners in class.</p>
---	---

Unit summary



In this Unit, we have touched generally on the types of programming languages. Students learnt all basics about Internet Protocols, focusing on the Hyper Text Transfer Protocol (HTTP), and the different methods by which certain servers are able to perform requests over the Internet.



Review Questions



1. Explain what you understand the term “Web Communication Protocol”. Explain how it works and give one example.
2. What is Web browser? Mention three examples, explain their differences.
3. Explain the meaning of URL and mention its basic parts.
4. Define the following terms and give two examples for each:
 - Domain identifier
 - Internet Protocol
 - Web browser
 - Web server
5. HTTP is stateless. What does this mean?
6. Describe the major difference between HTTP 1.0 and HTTP 1.1?
7. What protocol allows multiple objects to be transferred within one TCP connection?

Reference and Further Reading



1. Ada Programming/Pragmas/Atomic. (n.d.). Retrieved September 14, 2016, from <https://en.wikibooks.org/w/index.php?oldid=221748>
2. Hypertext Transfer Protocol. (2016, September 2). Retrieved September 14, 2016, from https://simple.wikipedia.org/wiki/Hypertext_Transfer_Protocol
3. Communication protocol. (2014, March 12). Retrieved September 14, 2016, from http://wiki.kidzsearch.com/wiki/Communication_protocol



4. Diploma Programme - computer sciences Wiki - C.4.2 cloud computing vs. Client-server architecture. Retrieved February 27, 2017, from <http://dis-dpcs.wikispaces.com/C.4.2+Cloud+computing+vs.+client-server+architecture>

Attribution

This unit of *Server Architecture and Web Protocols* is a derivative copy of materials from [Client Server Model](#) licensed under [Creative Commons Attribution License 3.0 license](#)

The following material was adapted from the link:

1. Key Terms
2. Notes



Unit 2

Introduction to Sever Side Scripting and Technologies

Introduction

A programming language is an artificial language that can be used to instruct a computer to do a particular task. To be considered a general programming language, it must be computationally complete, or Turing-Complete. It is nevertheless common to regard some languages that are not computationally complete, like database query languages and other domain-specific languages as programming languages as well. Programming languages enable you to create custom applications and add functionality that is not already part of an application. On the Internet, programming languages enable you to create visual animation, respond to user actions, validate forms, interact with databases, and provide e-commerce solutions.

In this module students will learn different web programming languages such as Perl, ASP, PHP, JSP, Cold Fusion and many more specialized choices. They will learn how to choose a platform with which to build database-driven Web application.

Unit outcomes



Upon completion of this unit you will be able to:

- Differentiate PHP to other Server-Side scripting Languages
- Differentiate CGI to other Server-Side scripting Languages
- Differentiate Perl to other Server-Side scripting Languages
- Differentiate JSP to other Server-Side scripting Languages
- Differentiate ASP to other Server-Side scripting Languages
- Differentiate NET Python to other Server-Side scripting Languages
- Differentiate Cold Fusion to other Server-Side scripting Languages



Terminologies



Server-Side Scripting:	A technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website
Database-Driven Web:	A website that has most of its webpage content stored in a database.
Frameworks:	Structure indicating what kind of programs can or should be built and how they would interrelate
Interpreter:	Computer program that directly executes instructions written in a scripting language, without previously compiling them into a machine language program.
Compiler:	Computer program that transforms source code written in a programming language into another computer language
Dynamic Websites:	A website written using a server-side scripting language such as PHP, ASP, JSP, or Coldfusion
Script Language:	Programming language that supports scripts, programs written for a special run-time environment that automate the execution of tasks
Server-Side Scripting:	A technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website

2.1 How to Choose Web Programming Languages

When it comes to the perfect programming language for the development of your site, it is imperative that you understand that there is no perfect programming language. Once you understand this, it is simply a matter of choosing the language that best serves your needs. Before you decide on what language to use, you should consider the following:

- Your server platform
- The server software you run
- Your budget



- Previous experience in programming
- The database you have chosen for your backend

2.2 List of Popular Programming Languages

2.2.1 Java/Servlet

Java is a powerful, open source, robust and secure web based program that can run as standalone program or an applet embedded in a website.

Servlets (controller), *JSP* (view), *Java* (model) creates web based applications using *MVC* (Model–view–controller). For creating robust secure web application *Java* has many frameworks like *Apache Struts*, *Spring* and *web_application_frameworks* .

Java Example Code

```
classHelloTest {  
    public static void main(String[] args) {  
        System.out.println("Content-type: text/html\n");  
        System.out.println("Hello World!"); //Display the  
        string.  
    }  
}
```

2.2.2 Java Server Pages (JSPs)

JSP are web pages with embedded *Java* code. The embedded *Java* code is executed on the server, and then the page is returned to the browser for display. The code below illustrates the structure of *JSP* script.

```
<html>  
  <head>  
    <title>A JSP Example</title>  
  </head>  
  <body>  
    <% out.println("<p>Hello there!</p>"); %>  
  </body>  
</html>
```



2.2.3 Perl

Perl was introduced in 1987 (4 years before Linux itself), when the author, **Larry Wall**, released version 1.000 of it. The reason for its creation was that Wall was unhappy by the functionality that sed, C, awk and the Bourne Shell offered him. He looked for a language that will combine all of their best features, while having as few disadvantages of its own.

Since then, perl has seen several versions, each adding additional functionality. perl version 5, which was released in 1994, was a complete re-write of the perl interpreter, and introduced such things as hard references, modules, objects and lexical scoping. Several second-digit versions of perl 5 appeared since then, and the most up-to-date stable version (as of November 2010) is 5.12.x.

Perl became especially popular as a language for writing server-side scripts for web-servers. But that's not the only use of perl, as it is commonly used for system administration tasks, managing database data, as well as writing GUI applications. The code below illustrates the basic structure of Perl script.

```
#!/usr/bin/env perl
use Dancer;
get '/' => sub
{
    "Hello World!"
};
dance;
```




2.2.4 PHP

PHP is a recursive acronym for PHP Hypertext Processor. Unlike the other offerings listed, it is designed specifically for server side programming, which means that its library is specialized for the tasks you'll be doing over and over again in the course of programming your website. PHP also has the advantage of being able to interweave code with HTML, thus allowing you to mix layout with programming. While this may simplify coding for small sites, it does carry the potential to be abused, making it difficult to manage or maintain code for larger projects. PHP is available for most operating systems including Unix and Windows, and is an excellent server side programming language for professional programming. The code below illustrates the basic structure of PHP script.

```
<!DOCTYPE html>
<html>
  <head>
    <title>PHP Hello World</title>
  </head>
  <body>
    <?php echo "hello world"; ?>
  </body>
</html>
```



2.2.5 C Language

The C programming language is a standardized programming language developed in the early 1970s by Dennis Ritchie for use on the UNIX operating system. It has since spread to many other operating systems, and is one of the most widely used programming languages. C is prized for its efficiency, and is the most popular programming language for writing system software, though it is also used for writing applications.

C is a useful language to learn since similar syntax is used by many modern languages such as Java, PHP, and JavaScript. The following piece of code illustrates the basic structure of C programming language:-

```
#include <stdio.h>
int main() {
    printf("Content-Type: text/html\r\n\r\n");
    printf("<html> <head>\n");
    printf("<title>Hello, World!</title>\n");
    printf("</head>\n");
    printf("<body>\n");
    printf("<h1>Hello, World!</h1>\n");
    printf("</body> </html>\n");
}
```

2.2.6 ASP

ASP, or Active Server Pages combine html and server side code (processed on the server and displayed as html) to create web pages with embedded functionality and scripts (see example code below).

```
<html>
<body>
<% dim name
name="Mathias Ombeni"
response.write ("My name is: " &name)
%>
</body>
</html>
```



2.2.7 ColdFusion

ColdFusion is a scripting language based on standard HTML that is used to write dynamic Web sites. It allows you create dynamic pages quickly and easily, including querying data from a database, use hundreds of built in tags and functions, or creating full scale object oriented enterprise level applications. ColdFusion pages can consist largely of standard HTML tags intermingled with ColdFusion Markup Language (CFML) tags, or can implement custom frameworks designed to assist the developer in separating presentation from business logic (e.g. with the MVC design pattern). ColdFusion was introduced by Allaire in 1996 and acquired by Macromedia in a merger in April 2001 and by Adobe in 2005. As an Adobe product, ColdFusion developers have the advantage of leveraging many existing Adobe technologies seamlessly.

Examples of such integration can be seen in the dynamic generation of Adobe Acrobat .pdf documents, Adobe Flash forms and presentations, Adobe Flash remoting capabilities, as well as connection with Adobe Flex user interfaces.

2.2.8 AJAX

AJAX is a short form of Asynchronous JavaScript and XML

AJAX is not a new programming language, but a new technique for creating better, faster, and more interactive web applications.

Ajax isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways. Ajax incorporates:

- Standards-based presentation using XHTML and CSS;
- Dynamic display and interaction using the Document Object Model;
- Data interchange and manipulation using XML and XSLT;
- Asynchronous data retrieval using XMLHttpRequest;
- And javascript binding everything together.



With AJAX, a JavaScript can communicate directly with the server, with the XMLHttpRequest object. With this object, a JavaScript can trade data with a web server, without reloading the page (see example code below).

```
<cfset myVar = "Hello World!">

<cfoutput>
  <html>
    <body>
      #myVar#
    </body>
  </html>
</cfoutput>
```

2.2.9 jQuery

jQuery is a client-side JavaScript library, the goal of the library is to simplify the process of writing cross-browser JavaScript code. jQuery was created by John Resig and it was released to the public in 2006, jQuery is free, open-source and is dual-licensed under the MIT license & GNU GPL Version 2.

jQuery's syntax is designed such that it enables developers to navigate DOM element with ease, assign & handle events easily without writing JavaScript code into HTML, like the onClick attribute. The Ajax functions provided by jQuery makes cross-browser Ajax applications very easy to write & takes off the headache to maintain compatibility with each and every browser. Below we'll look at a few examples of jQuery vs native JavaScript approach.

2.2.10 Common Gateway Interface (CGI)

Common Gateway Interface (CGI) is a standard way for web servers to interface with executable programs installed on a server that generate web pages dynamically. Such programs are known as CGI scripts or simply CGIs; they are usually written in a scripting language, but can be written in any programming language.



Video Lecture

<https://tinyurl.com/k3832kn>



<https://tinyurl.com/ktrjjzb>





Activity



Activity 2.0

Aim: Describe the difference between available server-side Technologies

Motivation: To become conversant with Sever side languages and technologies

Resources: Unit 2 notes

What to do: Using knowledge acquired from unit 2 notes, provide answers to the questions below:

1. Give a detailed comparison of Server-side Scripting Languages
2. Explain relative features, benefits and downfalls of all the major server-side development options available today
3. What is the best server side programming technology to work with JavaScript as a client-side and MySQL as a database?

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity to the course instructor and get a feedback on this activity.



Unit summary



In this module we have touched generally on the types of programming languages. Students learnt how to differentiate web programming languages such as Perl, ASP, PHP, JSP, Cold Fusion and many more specialized choices abound. They learnt how to Choose a platform with which to build database-driven Web application.

Review Questions



1. In reference to web programming, answer the following questions.
 - a) Point out with explanations Client-side scripting languages
 - b) What do you understand by the term server-side scripting? Explain and give three examples.
 - c) Explain what you understand by the term Hypertext Preprocessor (PHP)
2. List four (4) different languages for server-side scripting. Differentiate client-side and server-side scripting and explain how each of them can be used for validation



Reference and Further Reading

1. https://en.wikipedia.org/wiki/Comparison_of_web_frameworks
2. Web development. (2015, January 11) . Retrieved September 14, 2016, from https://en.wikibooks.org/wiki/Web_development
3. Common Gateway Interface. (2016, August 24). . Retrieved September 14, 2016, from https://en.wikipedia.org/wiki/Common_Gateway_Interface
4. Programming Languages/Introduction. (2014, November 23). . Retrieved September 14, 2016, from https://en.wikibooks.org/wiki/Programming_Languages/Introduction
5. AJAX jQuery Tutorial - An Introduction. Retrieved September 19, 2016, from <http://www.go4expert.com/articles/ajax-jquery-tutorial-introduction-t27338/>
6. Web development/Choosing the right programming language. Retrieved September 2, 2016, from https://en.wikibooks.org/wiki/Web_development/Choosing_the_right_programming_language



Attribution

This unit of *Sever Side Scripting and Technologies* is a derivative copy of materials from [choosing the right programming language](#) licensed under [Creative Commons Attribution-ShareAlike License](#).

The following material was adapted from the link

1. Notes



Unit 3

PHP Scripting Language

Introduction

In this Unit students will learn how PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them. Students will learn how PHP handles forms, i.e. gather data from files, save data to a file. They will learn how data are sent to the database and returned to the user using PHP language.

Unit outcomes

Upon completion of this Unit you should be able to:



- Describe the the basic structure of a PHP Script
- Identify PHP Error Messages
- Differentiate PHP Functions and Objects
- Write a simple PHP script
- Develop application using PHP script

Terminologies



Scripting Language:	Programming language that supports scripts, programs written for a special run-time environment that automate the execution of tasks
Variable:	Named memory location used to hold value
Operators:	Symbol or letter used to represent a certain operations on operands
Control Structure:	Determine the order in which individual statements, instructions or program are executed.
Functions:	Block of codes that performs specific tasks
Data Type:	A particular kind of data item, as defined by the values it can take



Local Variable:	Variable defined within a function and can be used only by that particular function
Global Variable:	Variable defined at the global level and can be used any function

3.1 About PHP

PHP was initially developed by a Greenlandic Programmer named Rasmus Lerdorf to facilitate developing his own personal Web page. After a while the package became sophisticated enough that he thought it was worth sharing with other people, so he released his set of programs, which he called *Personal Home Page Tools*, in mid-1995. (This was the time when the Web was getting its first widespread exposure and was becoming a mainstream phenomenon.)

At that time, the package was not yet something we would term a full-fledged programming language, but others found the package useful, and some joined Lerdorf to continue enhancing it. Two Israeli programmers, Andi Gutmans and Zeev Suraski, were interested in developing a language specifically for Web programming, and they thought PHP would be a good starting point. They joined Lerdorf and together the team developed the package into what we know today.

Along the way, PHP became appropriate for large Web sites, such as commercial sites, rather than just for personal home pages, so they began to refer to the package as simply *PHP*, which sounds like it could have once stood for Personal Home Page, but officially it stands for PHP: Hypertext Processor. Yes, the first *P* in PHP stands for PHP: They don't take this abbreviation idea too seriously.

PHP's structure is heavily influenced by the C programming language, a general-purpose language that has existed since 1978 and has been used to develop many large programs, including UNIX and Microsoft Windows. C's popularity led many subsequent languages to use its same



general syntax, so that programmers familiar with C would quickly feel comfortable using the new language instead. Other C-based languages include C++, Java, Perl, JavaScript, and C#. Of course, all are fairly distinct languages, but since all of these languages have a similar core, you will find them fairly familiar after seeing PHP.

PHP development is continuing, as those on the PHP team refine the language to enhance its security, facilitate development of large-scale Web sites, and provide other new features. As of mid-2007, when this was written, the current official version of PHP is version 5, although version 6 is nearing completion. Version 4 is still in wide use, partially because people haven't gotten around to installing the newer versions — but also because established Web sites need to take issues like security and reliability very seriously, and version 4 is a more thoroughly tested software package.

3.2 Variable and Data Types in PHP

3.2.1 Data types

These are the main types of variables you will be dealing with:

- **Int** – Short for integer. Holds any integer, i.e. a whole numerical value (no decimal places—however, if an int is assigned a number with a decimal place, it will be converted to a float automatically)
- **Float** – Abbreviation of floating point value. Holds any integer *with* a decimal place, to a certain precision (generally 32 digits)
- **Double** – Abbreviation of double precision floating point. Same as above except with far more precision—however, a double value also takes up twice as much memory (generally 8 bytes—obviously insignificant on modern servers)
- **String** – Holds any array of characters, i.e. a sentence (or two). Generally has an infinite length; however, for compatibility with older servers, limit the size of strings to 32768 characters. Strings are surrounded with single quotes when used in PHP.



- **Boolean** – Can either be true or false, many functions return true or false

3.2.2 Declaration

In a lower-level language like [C++](#), the declaration of the type of each variable is needed. For example, the following code in PHP:

```
<?php
$a='Stuff';
?>
```

Does not explicitly define the type of variable *a*. Of course, the PHP runtime environment recognizes during execution of this line that the programmer provided a string literal as the value for the variable. If you want to do the same in [C++](#), you would have to explicitly declare the variable type:

```
char[6] a;
```

```
a = "Stuff";
```

Additional equivalent statements combining declaration and initialization of a variable with the same value would be:

```
char* a = "Stuff"; // using char pointer
```

```
char b[] = "Stuff"; // using char array with string literal initialization
```

```
char c[] = { 'S', 't', 'u', 'f', 'f', '\0' }; // using char array with array initialization
```

Notice that C++ requires explicit variable declaration (including variable type) whereas PHP does not. However, it is good practise to declare all variables, as if they are not declared PHP defaults their value to FALSE (0, empty etc.). This is because PHP is a dynamically typed language, and as a result neither type casting nor declaration is enforced. The following is possible:

```
<?php
$a;
?>
```



3.2.3 Scope

Scope refers to what can actually modify a variable, e.g:

```
<?php
$var=1;
require'another_page.php';
?>
```

The page 'another_page.php' has full access to the variable 'var'.

```
<?php
$var='Test';
functiontest()
{
echo$var;
}
test();
?>
```

OUTPUT-> The above would output nothing, since functions have their own scope, variables outside that scope would not be available.

3.2.4 Globals

You can force functions to access other variables by using the global keyword, e.g.:

```
<?php
$var='Test';
functiontest()
{
global$var;
echo$var;
}
test();
?>
```

Would output: Test



Or you can use the \$GLOBALS array, like this:

```
<?php
$GLOBALS['var'] = 'Test';
function test()
{
    echo $GLOBALS['var'];
}
test();
?>
```

3.2.5 Static Variable

Static variables exist within a function and are saved after the function executes, e.g.:

```
<?php
function increment()
{
    static $a=0;
    echo $a;
    $a++;
}
increment(); // 0
increment(); // 1
increment(); // 2
increment(); // 3
?>
```

3.3 Operators

We have already seen how to use functions, such as `sqrt`, to manipulate values in PHP. For the most common operations, though, PHP provides symbolic shortcuts for the operations, such as `+` for adding two values together, or `*` for multiplying two values. Such symbols are called operators. PHP includes about 30 operators, though we'll only cover the



10 or so that appear frequently in programs. The most recognizable would be `+` for addition, `-` for subtraction or negation, `*` for multiplication, and `/` for division. We can use these to enhance our range selection page so that it displays the midpoint in the range

A. Assignment

1. `=` `+=` `-=` `/=` `*=` `%=` `++` `--` - like most programming language

2. `.` - string concatenation operator (see strings section)

B. Arithmetic

1. `+` `-` `*` `/` `%` - like most programming languages

C. Comparison

1. `==` `!=` `<>` `<=` `>=` - like most programming languages. Also `<>` is the same as `!=`.

2. `===` - true if arguments are equal and the same data type.

3. `!==` - true if arguments are not equal or they are not of the same data type.

D. Logical

1. `&&`, `||` - Like most programming languages (`&&` and `||` short-circuit)

2. `and`, `or` - like `&&` and `||` but have lower precedence than `&&` and `||`.

3. `xor` - true if either (but not both) of its arguments are true

3.4 Input Output

A. `print` and `echo` are used to print to the browser.

```
echo "Go Bisons";
```

```
echo("Go Bisons"); // same thing
```

```
print("Go Bisons"); // same thing
```




B. Print can only accept one argument, and echo can accept any number of arguments. Print returns a value that indicates if the print statement succeeded.

C. Variables are interpolated inside of strings unless single quotes are used.

```
$a = "guts";  
echo "You have $a."; // prints "You have guts."  
echo 'You have $a.'; // prints "You have $a."
```

D. Escape sequences: \n (newline), \r (carriage-return), \t (tab), \\$ (\$), \' ('), \\ (\)

```
echo "a\b\tc\d"; // prints "a\b c\d"  
echo 'a\b\tc\d'; // prints "a\b\tc\d". Only \\ is converted.
```

E. printf works like C's counter-part.

```
$title = "X-Men"; $amount = 54.235;  
printf("The movie <b>%s</b> made %2.2f million.", $title, $amount); //  
prints "The movie <b>X-Men</b> made 54.23 million."
```

F. PHP typically does not run from the command-line, but input from the keyboard can be accessed using the fopen function with "php://stdin". See the file I/O section for more information.

G. Output shortcut from within HTML:

```
Hello, <b><?=$name ?></b> is the same as Hello, <b><?php echo  
$name ?></b>
```



3.5 Control Structures

A. Choice structures

1. if ($\$x > 0$)

```
\$y = 5; // {} not required for only one statement
```

2. if ($\$a$) { // tests if $\$a$ is true or non-zero or a non-empty string print($\$b$);

```
\$b++; }
```

else

```
print(\$c);
```

3. if ($\$a > \b)

```
print "a is bigger than b";
```

```
elseif (\$a == \$b) // use "elseif" or "else if"
```

```
print "a is equal to b";
```

else

```
print "a is smaller than b";
```

4. switch ($\$vehicle_type$) { // works for integers, floats, or strings

```
case "car":    \$car++; break;
```

```
case "truck": \$truck++;    break;
```

```
case "suv":   \$suv++; break;
```

```
default: \$other++;
```

```
}
```

B. Looping structures

1. while ($\$n < 10$)

```
{
```

```
print("\$n "); \$n++;
```



```
}  
  
2.do {  
    print("$n "); $n++;  
} while ($n < 10)  
  
3.for ($n = 1; $n < 10; $n++) print("$n ");  
  
4.foreach ($myarray as $item) print("$item ");
```

3.6 Arrays

An array in PHP is a powerful structure for a script to remember a conglomeration of data. You should think of an as a collection of associations between *keys* and their corresponding *values*. For example: I might want to have an array that associates people's names with their addresses. We might choose the address to be the key, and the name to be the value associated with each key. Thus, associated with the key "221B" I would find the value "Sherlock Holmes." (If you happen to have seen the *array* notion in a different programming language, forget what you learned: PHP's notion of array is rather unusual.) An array is referenced using a single variable, such as `$residents`. To retrieve the value associated with a key, place the value into brackets following the variable name. Thus, given `echo $residents["221B"];` PHP will echo "Sherlock Holmes." We place a value into the array using this same bracket notation, but now on the left sign of an assignment statement. Thus, it would be legal to write `$residents["220A"] = "Watson";`. If `$residents` wasn't yet referring to anything, this would create an array. If the key 220A didn't already exist in the array, the key would be inserted with Watson as the associated value. And if there were already a value associated with 220A, that person would be evicted in favor of Watson. In fact, PHP automatically constructs some built in variables referring to arrays. The `$_POST` variable is one major example: When a browser sends a POST request to a PHP script, PHP sets `$_POST` to be an array whose keys correspond to the names of the input controls, and whose values are the values sent by the browser for each input control.



A. Summary of all array functions in the PHP core:

B. Arrays can have any size and contain any type of value. No danger of going beyond array bounds.

```
$my_array[0] = 25; $my_array[1] = "Bisons";
```

C. PHP arrays are associative arrays which allow element values to be stored in relation to a key value rather than a strict linear index order.

```
$capitals["CO"] = "Denver"; $capitals["AR"] = "Little Rock";
```

D. Initialize an array:

```
$colors =array ("red", "green", "blue");
```

```
print ("The 2nd color is $colors[1].");// prints      green
```

```
$capitals= array("CO" => "Denver", "AR" =>"Little  Rock");
```

```
print("$capitals[CO]") // prints Denver, no quotes around key inside ""
```

E. Print contents of an array for debugging:

```
print_r($colors);
```

produces:

Array

```
( [0] => red
```

```
[1] => green
```

```
[2] => blue
```

```
)
```

```
print_r($capitals);
```

produces:

Array

```
(
```



```
[CO] => Denver
```

```
[AR] => Little Rock
```

```
)
```

F. Pull values out of an array:

```
$colors = array("red", "green", "blue");
```

```
list($c1, $c2) = $colors;
```

```
print("$c1 and $c2"); // prints "red and green"
```

G. Delete from an array:

```
unset($colors[1]); // $colors now contains red and blue at indexes 0 and 2.
```

H. Extracting array keys and values:

```
$states=array_keys($capitals); //
```

```
$states is ("CO", "AR")
```

```
$cities = array_values($capitals);//
```

```
$cities is ("Denver", "Little Rock")
```

I. Iterating through an array:

```
$heroes= array('Spider-Man', 'Hulk', 'Wolverine');
```

```
foreach ($heroes as $name)
```

```
print("$name<br />"); // prints all three in order
```

```
foreach ($capitals as $state => $city) print("$city is the capital of $state.<br />");
```

J. Treat an array like a stack:

```
array_push($heroes, 'Iron Man'); $heroes[] = 'Captain America'; $h =  
array_pop($heroes);
```

```
//Pushed onto end of array
```

```
//Same thing as array_push
```



```
//Pops off last element (Iron Man)
```

K. Size of an array:

```
$num_items = count ($heroes); // returns 3
```

L. Sort an array:

```
sort($heroes); // Heroes are now in alphabetical order (lowest to highest)
```

```
rsort($heroes); // Reverse alphabetical order (highest to lowest)
```

3.7 Functions

Functions (or methods in the context of a class/object) are a way to group common tasks or calculations to be re-used simply. Functions in computer programming are much like mathematical functions: You can give the function values to work with and get a result without having to do any calculations yourself.

How to call a function

Note that echo is not a function. "Calling a function" means causing a particular function to run at a particular point in the script. The basic ways to call a function include:

- calling the function to write on a new line (such as after a ";" or "{")

```
print('I am human, I am.');
```

- calling the function to write on a new line inside a control

```
if ($a==72) {  
print('I am human, I am.');
```

- assigning the returned value of a function to a variable "\$var = function()"

```
$result= sum ($a, 5);
```



- calling a function inside the argument parentheses (expression) of a control

```
while ($i < count($one)) {  
  
}
```

In our earlier examples we have called several functions. Most commonly we have called the function `print()` to print text to the output. The parameter for `echo` has been the string we wanted printed (for example `print("Hello World!")` prints "Hello World!" to the output).

If the function returns some information, we assign it to a variable with a simple assignment operator "=":

```
$var1=func_name();
```

Example

A function declaration can look like this:

```
function print_two_strings($var1, $var2) {  
    echo $var1;  
    echo "\n";  
    echo $var2;  
    return NULL;  
}
```

To call this function, you must give the parameters a value. It doesn't matter what the value is, as long as there is one:

```
print_two_strings("Hello", "World");
```

Output:

```
Hello
```

```
World
```



3.8 Strings

PHP includes many other types for values that aren't numbers. One of the most important such is the string, which is a value representing a sequence of several characters, such as a word, a sentence, or a nonsensical stream of punctuation.

To refer to a particular string value, you can use a set of quotation marks enclosing the characters you want to include in the string. The following statement makes the variable `$sentence` refer to a famed novel's opening sentence.

```
$name = "It was the best of times; it was the worst of times.";
```

When you enclose a string in quotation marks, the contents of the quotation marks are not interpreted by PHP: It simply places the raw characters into the string. Let us look at the following code;

```
$expr = "rand(1, 100)";
```

PHP does not use the `rand` function here: It simply creates a string consisting of 12 characters, starting with the letter `r` and ending with the close-parenthesis character. If we omitted the quotation marks, of course PHP would call the `rand` function to retrieve a random number between 1 and 100.

One exception (and the only exception) to this rule are variable names: When you include a variable name in a string, PHP will substitute the variable's value in place of the name. Suppose I write this:

```
echo "The range is $form_begin to $form_end. ";  
echo "The midpoint is ($form_begin + $form_end) / 2. ";  
echo "The chosen number is rand($form_begin, $form_end). ";
```

The PHP interpreter would substitute the values of the variables whenever they occur; but it would not attempt to apply the operators or call the functions. What we would see on the Web browser, then, is the following.



The range is 10 to 20. The midpoint is $(10 + 20) / 2$. The chosen number is `rand(10, 20)`.

Of course, that's probably not what we want. There are several ways of getting around this. One is to first assign a variable to refer to the desired value, and then to use that variable inside the string. But another approach worth knowing about is the **catenation operator** `'.'`, which you can use to join two values into a string. The following is how we could modify our example.

```
echo "The range is $form_begin to $form_end. ";  
echo "The midpoint is ". (($form_begin + $form_end) / 2) . ". ";  
echo "The chosen number is " . rand($form_begin, $form_end) . ". ";
```

As an example, the final line says to catenate the string The chosen number is with the value generated by `rand`, which itself should be catenated with the string `..`. Note the space that has carefully been added inside the first string after `is`: Without this, the output would omit the space, as in the chosen number is17. The catenation operator has the same priority level as addition and subtraction. As a result, the following would be incorrect.

```
echo "The sum is " . $form_begin + $form_end; //WRONG!
```

PHP would first catenate The sum is with `$form_begin`, resulting in a string such as The sum is 20. Then it would attempt to interpret this as a number so that it could add `$form_end` to it. It would not work as we would expect. For this reason, I recommend developing the habit of using parentheses whenever you try to combine other operators with catenation, even when it is not necessary. The above midpoint example illustrates this: Even though there the parentheses were unnecessary since division occurs at a higher level than catenation, I went ahead and used parentheses anyway.

We've seen that variable names in double quotation marks will be replaced with the variable values. But that brings up a question: What if you don't want this substitution to occur? There are two answers that PHP



provides. First, you can prefix the relevant dollar sign with a backslash, as in "`\$choice is $choice`"; in this example, the first dollar sign won't be treated as part of a variable name, but the second will, so the string will translate to something like `$choice is 17`. You can do the same backslash trick when you want to a quotation mark to appear inside the string, or when you want a backslash as part of the string.

Backslashes can also be used to insert other characters. Most significantly, `'\n'` inserts a **newline character** into a string. Since line breaks aren't automatically inserted between `echo` statements, a PHP script that echoes a large amount of information would normally create one large line, which can be quite difficult to read for somebody who tries to read the generated HTML source.

3.9 Regular Expressions

A regular expression is a common way of describing a large set of possible strings. It can be seen as a miniature language, although it is expressed in a condensed form. Regular expressions represent a way to identify *patterns* in a text. They can be used to search for patterns and replace them with text, or with different patterns. They can also be used to identify a piece of a text for special handling.

Dots, stars, plusses, and backslashes

In regular expression syntax, a dot means "any single character." So the regular expression `b.b` will match "bib", "bob", "brb", or "bub". It will also match two b's with a space or a tab between them; the dot matches these "whitespace" characters also.

(Quick exercise: Where would the pattern `b.b` match in the previous sentence?).

This is helpful, but not quite enough. After all, there might be any number of letters between "Dear" and the colon in our salutation. Regular



expression syntax offers three ways to say “not just one”: the question mark (?), the star (*) and the plus sign (+).

The question mark means “zero or one of the previous character.” So the regex `Bb?` will match “B” (one capital B, zero lowercase b) or “Bb” (one capital B, one lowercase b).

The star means “zero or more of the previous character.” So the regex `Bb*` will match “B”, “Bb”, “Bbb”, “Bbbb”, and so on. (Note that you must have the capital B, or the match will fail; the B must match before `b*` can try to.)

The plus means “one or more of the previous character.” So the regex `Bb+` will match “Bb”, “Bbb”, “Bbbb”, and so on, but will *not* match “B” by itself.

Combining the dot with the plus or star solves

Cancelling special meanings

But what if you want to find an actual period (or an actual plus, or an actual question mark, or an actual star) in a regular expression? What if (to use a silly example) you want to find every letter where you mistyped the colon as a period in your salutation?

The backslash (\) signals to a regular expression that the following character, if it has a special regular expression meaning, should be interpreted literally, not in its special regex sense. So, by itself means “any single character,” while `\.` with the backslash means an actual period. This works for any character that has a special meaning in a regular expression: `*` means an actual star, and `\+` means an actual plus. Of course, the backslash works on itself, too; to search for a literal backslash, a regex pattern must contain `\\`.

Some other special regex characters

Aside from brackets and parentheses (to be discussed in future lessons), a few other characters have special meanings in a regex. They aren’t used as often as the dot, question mark, star, and plus; I mention them mostly so that you know to use a backslash if you mean the actual character.



- i. `\d` = any digit (0-9)
- ii. `\s` = any white space (space, tab, EOL) `\w` any word char
- iii. `(a-z, A-Z, 0-9, _)`
- iv. `.` = any character except EOL
- v. `[abc]` = a, b, or c
- vi. `[^a-z]` = not any char between a and z
- vii. `{3}` match only three of these
- viii. `?` match zero or one character
- ix. `*` match zero or more characters `+` match one or more characters
- x. `^abc` match at the beginning of the string `abc$` match at the end of the string.

Character classes

Remember the dateline of our letter? We could find it with the regex `..+ ..?,`, but that isn't terribly specific, and it looks awful. Could you easily guess, looking at that regex, that what you want is a date? Dates are far more predictable than that; they contain (ignoring spaces) a word, a number, a comma, and a four-digit number. Surely there is some way to distinguish letters from numbers?

Indeed there is. Regex engines allow you to create "character classes" that narrow a search to specific collections of characters. Character classes are contained within square brackets `[]`. Simply put the collection of characters you want to search for inside the square brackets. If you want not just one of them, you may use the question mark, dot, star, or plus after the ending square bracket. So one way to find our elusive dateline would be:

```
[JFMASOND][abceghilmnoprstuvy]+  
[123]?[0123456789], [12][09][0123456789][0123456789]
```



Fortunately, there is a shortcut. Square brackets also allow you to look for ranges of characters, such as “the digits 0 through 9” or “the lowercase letters a through z.” Simply separate the beginning and end of the range with a hyphen: for example, the character class for “all digits” is `[0-9]`.

Let’s try that dateline again:

```
[A-Z][A-z]+ [1-3]? [0-9], [12][09][0-9][0-9]
```

It could be simplified further at the cost of a little specificity, but as it stands, it’s fairly understandable, and will get the job done.

One thing you must know about character classes is that special regex characters are not special any more inside them. So another way to search for literal dots, stars, etc. is to put them inside square brackets: `[.*+]` would look for a dot, a question mark, a star, or a plus. Another handy tip involves hyphens: to include them in a character class (since normally they indicate a range), they must appear as the first character in that class. So the regex `[-0-9]` would search for any number or a hyphen.

One more important trick with character classes is that they are *negatable*: that is, you can just as easily look for any character that is *not* in a class, as any character that *is*. You could look for “any character that is not a number,” or “any character that is not a space.” To negate a character class, put `^` (shift-6) as the first character in the class. The regex `^[^0-9]` represents any non-number character (like `.` without the numbers).

But `^` already has a meaning! Yes, it does, but the meaning “at the beginning of a document” only applies *outside* character classes. So the expression `^[^A-Z]` will find a non-capital-letter character (a number, space, punctuation mark, or lowercase letter) at the very beginning of a document. Negation is truly handy when looking for something between *delimiters*, like parentheses, square brackets, or wedges. The easiest way to find something between wedges (such as an XML tag) is to type the opening wedge, then a class negating the closing wedge, then the closing wedge: `<[^>]+>`. It looks terrible, but works beautifully, and is much



more reliable than trying to use the dot (for reasons to be explored in the next lesson).



SUMMARY

Symbol	Meaning outside character class	Meaning inside character class
.	Any single character	.
?	Zero or one of the previous character.	?
*	Zero or more of the previous character.	*
+	One or more of the previous character.	+
^	Beginning of a document.	Negates character class.
\$	End of a document.	\$
[]	Demarcates a character class. To interpret literally, use a backslash.	N/A. Use a backslash for the literal characters.
()	Marks a group of characters to be remembered separately.	()
\	Forces a special character to be interpreted literally. Makes some ordinary characters special.	Forces a special character to be interpreted literally. Makes some ordinary characters special.
\n	Newline.	Newline.
\t	Tab.	Tab.
\s	Any whitespace character (including tabs and hard returns).	Any whitespace character (including tabs and hard returns).
\d	Any digit.	Any digit.

Video Lecture

<https://tinyurl.com/lz6wrw7>





<https://tinyurl.com/kvdxv2e>



<https://tinyurl.com/l34qjmg>



<https://tinyurl.com/m9ciclg>



<https://tinyurl.com/lkbm95n>





<https://tinyurl.com/llrdnu8>



<https://tinyurl.com/k4rvrr7>



<https://tinyurl.com/nx52nm4>



<https://tinyurl.com/kpgxaua>





Activity



Activity 3.0

Aim: Demonstrate the basics of PHP.

Motivation: To become conversant with basic PHP operations.

Resources: Unit 3 Notes

Tools: Notepad ++ or any related text editor and WAMPP or XAMMP Server

What to do:

- Write a PHP script to get the PHP version and configuration information.
- Write the phrase “Hello World” as HTML, with a line break afterwards
- Print the phrase “Strawberry Jam” using a php print statement
- Make “Hello World” bold 4. Make “Strawberry Jam” bold
- Make a variable called **firstname** and **age**, assign it the value of your own first name and age.
- On the next line, print a line that says “<name> is <age> years old.” and make the whole line italic where <name> if your **firstname** and <age> is **age** assigned

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity to course instructor for assessment and feedback.



Unit summary



In this Unit, we have covered about PHP language. PHP basically it is an HTML-embedded server side scripting language. In this Unit you learnt how PHP performs system functions, i.e. from files on a system to create, open, read, write, and close them. You learnt how PHP handles forms, i.e. gather data from files, save data to a file. You also learnt how data are sent to the database and returned to the user using PHP language.



Review Questions



1. What does PHP stand for?
2. How do you write "Hello World" in PHP?
3. Which symbol does all variables in PHP start with?
4. The PHP syntax is most similar to which language?
5. Explain two PHP super global used to collect form-data
6. Write down a function used to print statement in PHP.
7. What will be the output of PHP code below?

```
<? php  
  
define( ", "5");  
$x=x+10;  
echo x;  
?>
```

8. Write down a symbol which starts in all variables in PHP.

Reference and Further Reading



1. Chapter 5. Numbers and strings. (n.d.). Retrieved October 01, 2016, from <http://www.toves.org/books/php/ch05-types/index.html>
 2. PHP Programming. Retrieved October 01, 2016, from https://en.wikibooks.org/wiki/PHP_Programming
 3. Regular Expressions: A Brief Tutorial. (n.d.). Retrieved October 01, 2016, from <http://misc.yarinareth.net/regex.html>
-



Attribution

This unit of *Client PHP Scripting Language* is a derivative copy of materials from The Book [Programming via PHP](#) and [Regular Expressions](#) licensed under [Creative Commons Attribution License 3.0 license](#)

The following material was adapted from the link:

1. Notes
2. Activities



Unit 4

Interacting with Database

Introduction

This lecture gives a student a solid introduction to using MySQL database with PHP programming language to build database driven websites. Students will learn the SQL language and master database design principles. It provides the basic skills to create three-tiered data applications such as websites that require user login/authentication, websites with automated web content, interactive websites. To develop web applications that involve business logic and basic database operations, students will learn how to evaluate server-side scripting technology and develop server-side scripts using appropriate tools.

Unit outcomes



Upon completion of this unit you should be able to:

- Write codes used in PHP to connect to MySQL database
- Describe how the MySQL server can accessed.
- Explain how to Display Content in a Web Page
- Write codes to Query a MySQL Database with PHP
- Create a Database

Terminologies



Database:	Is a collection of information that is organized so that it can easily be accessed, managed, and updated
DBMS:	A system software for creating and managing databases
SQL:	Is a language for composing queries for a database



Table:	Database object used to data
Query:	Object used to ask specific information from a database
Database:	Is a collection of information that is organized so that it can easily be accessed, managed, and updated

4.1 Understand MySQL

One of the most common applications of PHP is to provide a Web interface for accessing a database. The database may hold information about user postings for a forum, account and order information for a vendor, or raw data for a statistics site.

Because databases are so common, there are a special categories of programs written specifically for managing databases, called database management systems (more often referenced by their abbreviation DBMS). Typical users don't often know that they're dealing with DBMSs, so the names of specific products aren't widely known. Some of the more popular commercial DBMS packages are Oracle and Microsoft SQL Server, but there are also two prominent open source DBMS packages, MySQL and PostgreSQL.

We'll use MySQL in our study, since it is easily available and used quite frequently for Web pages in conjunction with PHP. In fact, PHP's interfaces for interacting with the different DBMSs are all quite similar, so if you wanted to use another, it wouldn't take much to learn.

Most DBMSs represent a database as a set of tables, each of which has a relatively fixed set of columns, and a more dynamic set of rows. A row represents a single entity, and a column represents an attribute of an entity. The simplest thing is to look at an example: Suppose we are managing a Web forum using a database. One thing we might want would be a table listing all participants in the forum. Each participant would have a distinctive user ID, a password, a real name, an e-mail



address, and the year the participant joined the forum. Each of these attributes will be a column in our table.

4.2 Simple SQL Retrieval

We will want to ask the DBMS to retrieve information for us. There is a language designed specifically for this, called SQL (an acronym for Structured Query Language). SQL is so widely popular that it is closely identified with high quality DBMSs — which is why so many of the popular DBMSs, including MySQL, incorporate the name SQL into their name.

SQL is a language for composing queries for a database. It is not a full-fledged programming language, but it is powerful enough to represent all of the typical operations on a database. PHP scripts send requests to the database using SQL, so we need to learn some of its fundamentals.

For now, we'll look only at the simplest version of the most important SQL command: the `SELECT` query. A simple `SELECT` query is composed of three separate clauses: a `SELECT` clause listing the names of the columns whose values we want to see, a `FROM` clause saying which table we want to access, and a `WHERE` clause indicating which rows we want to retrieve from that table.

For example, suppose we wanted to look up Sherlock Holmes' password and e-mail address. The SQL query we would want to send to the DBMS is the following.

```
SELECT passwd, email
FROM Users
WHERE name = 'Sherlock Holmes'
```

Note that the `WHERE` clause is a condition. In this case, we want to compare the name of each row to the string "Sherlock Holmes" (in SQL, strings are enclosed in single quotes). When we find a row that matches, we want the DBMS to send back the columns named in the `SELECT` clause.



The WHERE clause can include several conditions joined by AND and/or OR.

```
= SELECT name  
= FROM Users  
= WHERE year_joined > 1900 AND year_joined < 1970
```

If the table held the three people listed above, this SQL query would result in two values: Miss Marple and Nancy Drew.

4.3 PHP Database Functions

PHP provides access to [MySQLdatabases](#) via a number of functions. We'll find six of them particularly useful. They are listed below in the order they will typically be used.

```
$db = mysql_connect($dbms_location)
```

Connects to the DBMS whose address is identified by the parameter, which should be a string. The exact string will depend on where exactly the DBMS is located;

An example string is localhost:/export/mysql/mysql.sock.

The function returns a value that can later be used to refer to the DBMS connection.

```
mysql_select_db($db_name, $db)
```

Selects which database managed by the DBMS that this PHP connection will reference. The DBMS will have a name for each database, and the name should be passed as a string for the first parameter. The second parameter should be a reference to the DBMS connection as returned by [mysql_connect](#).

```
$result = mysql_query($sql, $db)
```



Sends a query to the DBMS. The SQL code should be located in the string passed as the first parameter; the second parameter should be a reference to the DBMS connection as returned by `mysql_connect`. The function `mysql_query` returns a value that allows access to whatever the results were of the query; if the query failed — as for example would happen if there were an error in the given SQL — this return value would be `FALSE`.

`mysql_error()`

If the last executed query failed for some reason, `mysql_error` returns a string that describes this error. I recommend that you always make sure your PHP code somehow echoes the result of `mysql_error` when a query fails, so that you have a way to debug your script.

`mysql_num_rows($result)`

Returns the number of rows found by the SQL query whose results are encapsulated in the parameter. Of course, this could be 0 if there were no rows that resulted from the query.

`mysql_result($result, $row, $col)`

Returns a string holding the information found in row `$row` and column `$col` in the results that are encapsulated by the `$result` parameter. Both rows and columns are numbered starting from 0: That is, the first row is numbered 0, the second row 1, and so on, and similarly the first column is numbered 0.



Example database access

Suppose we want to write a script that allows a user of our Web site to view the information about a particular participant. The Web site user would see a form such as the following.

User ID:

This form can be generated by the following HTML code.

```
<form method="post" action="user.php">  
<p><b>User ID:</b> <input type="text" name="userid" /></p>  
<p><input type="submit" value="Search" /></p>  
</form>
```

Video Lecture

<https://tinyurl.com/laz7hoe>





Activity



Activity 4.0

Aim: Demonstrate basics of MySQL.

Motivation: To become conversant with basic MySQL Statements and its operations.

Resources: Unit 4 Notes

Tools: Notepad ++ or any related text editor and WAMP/XAMMP MySQL server

What to do:

- Write a SQL statement to create a simple table countries including columns country_id, country_name and region_id.
- Write a SQL statement to insert a record with your own value into the table countries above against each columns.ndregion_id.
- Write a query to display the countries from table countries.

How to do it:

Read notes in Unit 4 and apply knowledge to do this activity

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity to the course instructor and get a feedback on this activity.

Activity 4.1

Aim: Demonstrate basics of MySQL.

Motivation: To become conversant with basic MySQL Statements and its operations.

Resources: Unit 4 Notes

Tools: Notepad ++ or any related text editor and WAMP/XAMMP MySQL server

What to do:

- Write a SQL statement to create a simple table countries including columns



country_id, country_name and region_id.

- Write a SQL statement to insert a record with your own value into the table countries above against each column's region_id.
- Write a query to display the countries from table countries.

How to do it:

Read notes in Unit 4 and apply knowledge to do this activity

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity to the course instructor and get a feedback on this activity.



Unit summary



In this unit you learned about database models and their categories. In addition, you have also learned about different types of database models which has formed the basis of databases in use in the industry.

In this lecture you were introduced to g MySQL database with PHP programming language and how to build database driven websites. Students learnt the SQL language and master database design principles. You learnt how to develop web applications that involved business logic and basic database operations, you learned how to evaluate server-side scripting technology and develop server-side scripts using appropriate tools.

Review questions



1. Explain the DBMS.
2. Briefly explain the relationship between PHP and MySQL
3. Explain the use of SELECT and WHERE.
4. Illustrate how you would connect to database using PHP
5. What is the purpose of the semicolon in MySQL queries?
6. Which command would you use to view the available databases or tables?
7. What is the purpose of a MySQL index?



Reference and Further Reading



1. Wikibooks, MySQL. Retrieved September 15, 2016, from <https://en.wikibooks.org/wiki/MySQL>
2. Burch, C. Chapter 7. Database access. Retrieved September 18, 2016, from <http://www.toves.org/books/php/ch07-db/index.html>

Attribution

This unit of *Interacting with Database* is a derivative copy of materials from [PHP and MySQL Programming/Database Connectivity](#) and [Database access](#) under [Creative Commons Attribution ShareAlike 3.0 International](#)

The following material was adapted from the link:

1. Notes
2. Activities



Unit 5

Sessions and Cookies in PHP

Introduction

The web is “stateless”... meaning the browser does not maintain a connection to the server while you are looking at a page. You may never come back to the same server or it may be a long time or it may be one second later.

In this Unit you will learn how to maintain state by using cookies and Sessions. You will learn how browser state is stored in cookies and how server state is stored in Sessions.

Unit outcomes



Upon completion of this unit you should be able to:

- Explain how ‘sessions’ work
- Write PHP codes illustrating starting or resuming a Session
- Explain how session data are stored
- Explain process of reading session data
- Understand how a session is destroyed

Terminologies



Cookies:	A small piece of data sent from a website and stored in the user's web browser while the user is browsing
Sessions:	A way to store information (in variables) to be used across multiple pages
Analytic:	The discovery, interpretation, and communication of meaningful patterns in data



5.1 Understand Sessions

5.1.1 Importance of Sessions

Classical server-side Web programs suffer many limitations: A Web server handles each Web request individually, without the context of other requests coming from the same user. This is a major limitation in situations such as sites that require users to log in and others where the user builds a "shopping cart" of desired items to buy. Such problems require that a server-side script "remember" what happened with previous requests, which can become a fairly complicated process as a Web server interleaves servicing requests from a variety of users.

PHP addresses this problem with the notion of a session. Using sessions with PHP in their basic form is quite simple: It is simply a matter of calling the `session_start` function before sending any HTML code to the client. PHP will create an array called `$_SESSION` corresponding to this user; anything placed in this array will be associated with the session, and will be recalled into the array when the same user later requests a script that also calls `session_start`. PHP does quite a bit behind the scenes to allow sessions to work well.

The following is a simple example involving sessions: It is a PHP script that counts how many times a user has loaded the page in question, it



illustrate all of the concepts essential to using sessions profitably.

```
1 <?php
2 session_start();
3 4
4 if(isset($_SESSION["count"])) {
5 $accesses = $_SESSION["count"] + 1;
6 } else {
7 $accesses = 1;
8 }
9
10 $_SESSION["count"] = $accesses;
11 ?>
12 <html>
13 <head>
14 <title>Access counter</title>
15 </head>
16
17 <body>
18 <h1>Access counter</h1>
19
20 <p>You have accessed this page <?php echo $accesses; ?> times
today.</p>
21
22 <p>[<a href="counter.php">Reload</a>]</p>
23 </body>
24 </html>
```

Source: <http://www.toves.org/books/php/ch14-sessions/>

5.2 Understand Cookies

Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. SilverStripe uses cookies for remembering users' preferences. Application code can modify users' cookies through the Cookie class. This class mostly follows the PHP API.

5.3 Uses of Cookies

Cookies are often used to perform the following tasks:

5.3.1 Session management:

Cookies are widely used to manage user sessions. For example, when you use an online shopping cart, you keep adding items in the cart and finally when you checkout, all of those items are added in the list of items you have purchased. This can be achieved using cookies.



5.3.2 User identification:

Once a user visits a webpage, using cookies, that user can be remembered. And later on, depending upon the search/visit pattern of the user, content which the user likely to be visited is served. A good example of this is 'Retargeting'. A concept used in online marketing, where depending upon the user's choice of content, advertisements of relevant product, which the user may buy, are served.

5.3.3 Tracking / Analytics:

Cookies are used to track the user. Which, in turn, is used to analyze and serve various kind of data of great value, like location, technologies (e.g. browser, OS) form where the user visited, how long (s)he stayed on various pages etc.

5.4 How to Create a Cookie in PHP

Setting a cookie

Setting a cookie is extremely easy with `setcookie()`

5.4.1 Syntax

```
boolsetcookie (stringname [, stringvalue [, intexpire [, stringpath [, stringdomain [, boolsecure]]]])
```

Where name is the cookie name, values the data to be contained in the cookie, expire the time after which the cookie should expire, path the path on the server which can use the cookie, domain can be used to set permissions for sub-domains and secure if set true only transmits the cookie if a secure connection is present.

Since all cookies are sent by the server along with HTTP headers you need to set any cookie at the start of a page **before** any other code. You will normally only need to use the name, value and expire arguments. If expire not set the cookie will expire when the client closes the browser.



Examples

```
setcookie("wikibooks", "user", time() + 3600);
```

The above code will set a cookie having the name wikibooks, value user and will expire an hour after it is set.

```
setcookie("test", "PHP-Hypertext-Preprocessor", time() + 60,
"/location", 1);
```

Here the setcookie function is being called with four arguments (setcookie has 1 more optional argument, not used here). In the above code, the first argument is the cookie name, the second argument is the cookie contents and the third argument is the time after which the cookie should expire in seconds (time() returns current time in seconds, there time()+60 is one minute from now). The path, or location, element may be omitted, but it does allow you to easily set cookies for all pages within a directory, although using this is not generally recommended.

You should note that since cookies are sent with the HTTP headers the code has to be at the top of the page (Yes, even above the DOCTYPE declaration). Any other place will generate an error.

5.4.2 Parameters

setcookie() has several parameters. Following table shows details of the parameters:

S/N	Parameter	Description	Data Type
1	name	Name of the cookie.	String
2	value	Value of the cookie, stored in clients computer.	String
3	expire	Unix timestamp, i.e. number of seconds since January 1st, 1970 (called as Unix Epoch).	Integer



4	path	Server path in which the cookie will be available.	String
5	domain	To which domain the cookie is available.	String
6	secure	If set true, the cookie is available over secure connection only.	Boolean
7	httponly	If set true, the cookie is available over HTTP protocol only. Scripting languages like JavaScript won't be able to access the cookie.	Boolean
8			

setcookie() returns boolean.

5.5 How to Delete Cookies

The `$_COOKIE` will returns all the cookies from domain and path. To delete the cookie with WordPress, we need to delete the cookie value and set the expire date to past. This will tell our browser to delete the expired cookie automatically.

```
Add_action('init','my_deletetecookie');
function my_deletetecookie()
{
    setcookie ('my-name','', time()-3600, COOKIEPATH.
    COOKIE_DOMAIN);
}
```

We can do cookie deletion by clearing our browser cookie, or by right click using Firebug (image above) to delete the selected cookie. Playing with cookie only need a few lines of code. We can create function to handle our need, tracking user visits, downloads and many other user defined tasks.



5.6 Retrieving cookie data

If a server has set a cookie on the user's computer, the user's browser sends it to the server each time a page loads. The name of each cookie sent by your server is stored in the 'superglobal array `$_COOKIE`'.

Considering the above example the cookie would be retrieved by calling `$_COOKIE['test']`.

To access data in the cookie we use *explode* (). *explode* () turns a string into an array with a certain delimiter present in the string. That is why we used those dashes(- hyphens) in the cookie contents. So to retrieve and print out the full form of PHP from the cookie we use the code:

```
$array=explode("-", $_COOKIE['test']); //retrieve contents of cookie  
print("PHP stands for ".$array[0] .".$array[1] .".$array[2]); //display the  
content
```

Note: `$_COOKIE` was introduced in 4.1.0. In earlier versions, use `$HTTP_COOKIE_VARS`.

5.7 Where are cookies used?

Cookies can be often used for:

- User preferences
- Inventories
- Quiz or poll results
- User authentication
- Remembering data over a longer period

You should **never** store unencrypted passwords in cookies as cookies can be easily read by other users. You should **never** store critical data in cookies as cookies can be easily removed or modified by other users.



Video Lecture

<https://tinyurl.com/mfsnw2l>



<https://tinyurl.com/kflvwz6>





Activity



Activity 5.0

Aim: Demonstrate basics of cookies and sessions functions.

Motivation: To become conversant with cookies function.

Resources: Unit 5 notes

What to do:

- Create a cookie named "username" and assign the value value "John Carter" to it, specify that the cookie will expire after 30 days (30 days * 24 hours * 60 min 60 sec).
- Use the PHP \$_COOKIE superglobal variable to retrieve a cookie value.
- use the PHP isset() function to check whether a cookie is set or not before accessing its value
- Use setcookie() function with the cookie name and any value to delete a cookie.

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity to the course instructor and get a feedback on this activity.

Unit summary



In this unit, the students learnt how to maintain state by using cookies and Sessions. They learnt how browser state is stored in cookies and how server state is stored in Sessions. They learnt how to create and destroy cookies in PHP.



Review Questions



1. What are Sessions?
2. How can you differentiate Sessions from Cookies
3. What is the importance of Cookies?
4. Identify PHP function that stores a cookie on a web browser
5. Explain how you can destroy a cookie, give an example

▪

Reference and Further Reading



1. Burch. C (n.d.). Chapter 1. Introduction. Retrieved September 18, 2016, from <http://www.toves.org/books/php/ch01-background/index.html>
2. How to Set and Get or Delete Cookies with WordPress - CodeCheese. Retrieved October 03, 2016, from <http://www.codecheese.com/2013/11/how-to-set-and-get-or-delete-cookies-with-wordpress/>
3. Wikibooks, PHP Programming/cookies. Retrieved October 03, 2016, from https://en.wikibooks.org/wiki/PHP_Programming/cookies

Attribution

This unit of *Sessions and Cookies in PHP* (including images, except as otherwise noted) is a derivative copy of materials from [Programming via PHP](#), licensed under [Creative Commons Attribution License 3.0 license](#)

The following material was adapted from the link:



1. Notes
2. Activities



Unit 6

Introduction to AJAX

Introduction

In this Unit you will be introduced in web development techniques used on the client –side to create asynchronous web applications. You will learn Ajax basics, dynamic display and interaction using the Document Object Model; data interchange and asynchronous data retrieval using XMLHttpRequest. You will learn how Ajax technologies can be used to send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behaviour of the existing page.

Unit outcomes



Upon completion of this unit you will be able to:

- Define Ajax and how it is implemented
- Explain the technologies combined in Ajax
- Explain how asynchronous processing is handled using Ajax
- Describe the formats and protocols used by AJAX

Terminologies



DOM	A programming interface for HTML and XML documents that provides a structured representation of the document
CSS	A style sheet language used for describing the presentation of a document written in a markup language.
Back-end Development	Building an application (using server-side code like PHP, Ruby, Python, .Net etc.) which connects with a database (using MySQL, SQL, Access etc.)



Front-end Development: The practice of producing HTML, CSS and JavaScript for a website or web application so that a user can see and interact with them directly.

6.1 AJAX Basics

Ajax isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways. Ajax incorporates:

- Standards-based presentation using XHTML and CSS;
- Dynamic display and interaction using the Document Object Model;
- Data interchange and manipulation using XML and XSLT;
- Asynchronous data retrieval using XMLHttpRequest;
- And JavaScript binding everything together. Ajax is based on open standards;

6.1.1 Asynchronous

Asynchronous in computer science is a form of sending/receiving data processing that permits other processing to continue before the transmission has finished.

6.1.2 JavaScript

JavaScript is used to make a request to the server, once the response is returned by the server. It was originally control the browser, communicate asynchronously and alter the document content that was displayed.

6.1.3 XML

The data that you receive back from the server will often be packaged up as a snippet of XML, so that it can be easily processed with JavaScript. This data can be anything you want, and as long as you want.



Ajax is based on open standards; AJAX is based on the following open standards; browser- based presentation using HTML and css. data stored in XML format and fetched from the server.

JavaScript/DOM. create XMLHttpRequest object to exchange data asynchronously with a server.

6.2 How Ajax works

Ajax request/response mechanism complete in three simple steps.

First step

- A user generates an event...
- Create an XMLHttpRequest object.
- Send HttpRequest .

Second step (server side)

- First process the HttpRequest.
- After create a response object and send data back to the browser.

Third step (browser side)

- Browser receives the data from server using JavaScript.
- The HTML DOM is updated.

6.3 Importance of Ajax

An Ajax application eliminates the start-stop-start-stop nature of interaction on the Web by introducing an intermediary — an Ajax engine — between the user and the server. It seems like adding a layer to the application would make it less responsive, but the opposite is true.

Instead of loading a webpage, at the start of the session, the browser loads an Ajax engine — written in JavaScript and usually tucked away in a hidden frame. This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf. The Ajax engine allows the user's interaction with the application to happen asynchronously — independent of communication with the server. So the user is never staring at a blank browser window and an hourglass icon



waiting around for the server to do something. The following are advantages of using Ajax:

- Ajax increase the web page's interactivity, speed, and usability.
- Stable, do not crash easily.
- Fast buffering of data.
- Secured transaction.

6.4 XMLHttpRequest Object

6.4.1 What is ajaxXMLHttpRequest object

The XMLHttpRequest object is the key to AJAX.

The XMLHttpRequest(XHR) is an API available in web browser scripting languages such as JavaScript. It is used to send HTTP or HTTPS requests directly to a web server and load the server response data directly back into the script. The data might be received from the server as JSON, XML HTML or as plain text.

All modern browsers support the XMLHttpRequest(xhr) object (IE5 and IE6 use ActiveXObject).

6.4.2 How to Create an XMLHttpRequest(XHR) Object

Internet explorer versions 5 and 6 did not define the XMLHttpRequest object its uses an ActiveX object. The syntax for creating an ActiveXObject :-

```
Var ref = new ActiveXObject("Microsoft.XMLHTTP");
```

All modern browsers such as Firefox, Safari, Opera, Chrome and IE7+ fully support the XMLHttpRequest object.

The Syntax for creating an XMLHttpRequest object.

```
Var ref = new XMLHttpRequest () ;
```

6.4 3 Browser Specific Codes.

Simply way to create your source code compatible to a browser is to use if... else if blocks in your javascript.



example

```
<script>
    //browser support code
function getajax() {
varajaxObject = null;
if (window.XMLHttpRequest) {
    //support IE7+, Firefox, Chrome, Safari, Opera.
    ajaxObject = new XMLHttpRequest ();
} else if (window.ActiveXObject) { // code for IE5 ,IE6.
    ajaxObject = new ActiveXObject("Microsoft.XMLHTTP");
    }
return ajaxObject;
}
```

</script>Send Ajax Request (XMLHttpRequest) To A Server

The XMLHttpRequestObject:- it is an API available in web browser scripting languages such as JavaScript. It is used to send HTTP or HTTPS requests directly to a web server and load the server response data directly back into the script.

The data might be received from the server as JSON, XML HTML or as plain text.

Data from the response can be used directly to alter the DOM of the currently active document in the browser window without loading a new web page document.

6.5 How to Send Request to a Sever

When we want to send a ajax request to a server, we use the open() and send() methods of the XMLHttpRequestobject .



The syntax:

Xmlhttp.send (content);

Xmlhttp.open (method,URL, async, userName, password);

Ajax Method	Description
send(content)	Send the request. Content indicate name/value pairs, pass with post request.
open(method,url,async)	<i>method</i> type of request methods GET,POST or HEAD. <i>URL</i> : indicate the location of file on the server. <i>Async</i> True : request generate as asynchronous type. False : request generate as synchronous type.
abort()	cancel the current request.
getAllResponseHeader s()	Returns the complete set of HTTP headers as a string.
getResponseHeader(he adername)	Returns the value of specified HTTP header.
setRequestHeader(labe l,value)	Set HTTP headers to the request. <i>Label</i> : indicate the header name value: indicate the header value.



6.6 Request Methods

The HTTP/1.0 specification defined the GET POST and HEAD methods. But over the internet only GET/POST method is useful. The GET method is simpler and faster than POST method.

6.6.1 GET requests:

The HTTP GET request method is designed to retrieve information from the server. If you want to send some information with the get method, add the information to the URL with (?) sign.

6.6.2 POST requests

POST is non-idempotent type method, we can sending large amount of data to the server because POST has no size limitations. The POST is more robust and secure than GET method.

6.6.3 The open () method description

The syntax:

```
XmlHttpRequest.open (method,URL,asynchronous);
```

Method parameter indicates the type of request, which can be GET or POST.

URL: open() method URL parameter is an address to a file on the server. The file can be any number of types such as .xml .text .asp .jsp .php .

Asynchronous -

```
XmlHttpRequest.open ("GET","signin.jsp",true);
```

When using asynchronous='true', assign a callback handler to the onreadystatechange property to determine when the call has completed.default is true.

```
XmlHttpRequest.open ("GET","signin.jsp",false);
```

When using asynchronous='false' send operations are synchronous. When a send operation is in progress, other operations are suspended.



Note: If you use `asynchronous='false'`, do NOT write an `onreadystatechange` function.

6.6.4. How do Ajax Respond

After a successful and completed call to the **send** method of the `XMLHttpRequest`, if the server response was well-formed XML and the *Content-Type* header sent by the server is understood by the user agent as an Internet media type for XML, the *responseXML* property of the `XMLHttpRequest` object will contain a DOM document object. Another property, *responseText* will contain the response of the server in plain text by a conforming user agent, regardless of whether or not it was understood as XML. The `responseText` or `responseXML` property of the `XMLHttpRequest` object use to get the response object from the server.

- The ajax response `responseText` property
- The `ajaxresponseText` property retrieves the response body as string.
- The `ajaxresponseText` property values type is string.

if `responseText` type is not the empty string or 'text', throw an `'InvalidStateError'` exception and terminate the process.

```
DOM element = object.responseText;
```

Example

ReponseText

The `responseXML` property

The `ajaxresponseXML` property retrieves the response body as an XML Document Object Model

(DOM)object.

The `ajaxresponseXML` property values type is object.

if `responseXML` type is not the empty string or 'document', throw an `'InvalidStateError'` exception and terminate the process.

```
xmlDocument = object.responseXML ;
```



Example

ResponseXML

The `responseBody` property. The `ajaxResponseBody` property retrieves the response body as an array of unsigned bytes. The `ajaxResponseBody` was introduced in windows internet explorer 7.

```
element = object.responseBody
```

The `ajax response` property. The `ajax response` property returns the response received from the server. This property is read-only. The `ajax response` received for the request or null if no response is received.

```
element = object.response
```

AJAX - onreadystatechange events

The `ajax - onreadystatechange` event is triggered every time the `readyStatechanges.event` handlers are called as needed after a request is sent. The `ajax onreadystatechange` event syntax :-

```
newXMLHttpRequest().onreadystatechange = handler ;
```

ajax - onreadystatechange event example :-

```
onreadystatechange
```

```
<script>
```

```
functionstatusHandler() {
```

```
if (requestObject.readyState == 4) {
```

```
if (requestObject.status == 200 || requestObject.status == 304) {
```

```
alert ("transfer completes") ;
```

```
    } else {
```

```
        //error occurred
```

```
    }
```

```
    }
```

```
    }
```

```
varrequestObject = new XMLHttpRequest();
```



```
requestObject.open("GET", "http://localhost/login.jsp", true);
requestObject.onreadystatechange= statusHandler ;
requestObject.send();
```

when the readyState is 4 and status is 200, the response is ready.

NOTE :- onreadystatechange event is triggered five times (0 to 4), one time for each change in readyState. onreadystatechange event handler a function to be called automatically each time the readyState property changes.

The readyState property:

The ajaxreadyState property defines the current state of the XMLHttpRequest object. The XMLHttpRequest object can be in several states, using readyState property retrieves the current state of the request operation, its return value type is Integer. The ajax readyState syntax :

object.readyState

The readyState property holds the status of the XMLHttpRequest.value can change from 0 to 4 :Here are possible values for the readyState property.

State Value	State name	State Description
0	Uninitialized	The object has been created, but not initialized.
1	Loading	open() method has been invoked, but send method has not been called.
2	Loaded	The send method has been called. no data is available yet.



3	interactive	Some data has been received however, neither responseText nor responseBody is available.
4	complete	All the data has been received.

AJAX status property

The ajax status property returns the HTTP status code.

The return value type is Integer.

```
var status = new XMLHttpRequest().status;
```

Some long integer status value is a standard HTTP status code as described in table.

Value	Description
100	Continue
101	Switching protocols
200	OK
202	Accepted
302	Found
404	Not Found

The statusText

The statusText property returns the HTTP status text

The return value type is String .



```
var status = new XMLHttpRequest().statusText;
```

Ajax - setrequestheader() method

the ajax - setRequestHeader() method adds custom HTTP headers to the request.

note :- The empty string is legal and represents the empty header value.

AJAX - setRequestHeader Method syntax

```
varreturnvalue =object.setRequestHeader(header , value);
```

parameter	description	type
header	Specifies the header name.	String
value	Specifies the header value.	String

ajax - setRequestHeader method example: the following script sets the HTTP content-Type header to "text/html" before sending the request body.

```
setRequestHeader
```

```
<script>
```

```
varxmlhttp = new XMLHttpRequest();
```

```
xmlhttp.open("GET", "../login.jsp", false);
```

```
xmlhttp.setRequestHeader("Content-Type", "text/html");
```

```
xmlhttp.send(requestBody);
```

```
</script>
```

HEAD part

```
HTTP/1.1 200 OK
```

```
Server: Microsoft-IIS/4.0
```

```
Cache-Control: max-age=212200
```

```
Expires: Tue, 12 apr 2012 11:34:01 GMT
```



Date: Mon, 05 May 2012 12:1:02 GMT

Content-Type: text/html

Accept-ranges: bytes

Last-Modified: thu, 15 mar 2012 12:03:20 GMT

ETag: "0a7ccac50cbc11:1aad"

Content-Length: 65332

AJAX - getResponseHeader() method

The ajax - getResponseHeader() method returns the specified response header.

The AJAX - getResponseHeader Method syntax

```
varreturn_value =object.getResponseHeader("header");
```

parameter	description	type
header	Specifies the response header name.	String
return_value	String that receives the response header value.	String

ajax - getResponseHeader method example: Using HEAD requests, to find the Last-Modified of another file.

```
getResponseHeader()

<script>

varxmlhttp = new XMLHttpRequest();

xmlhttp.open("HEAD", "../login.html",true);

xmlhttp.send();

xmlhttp.onreadystatechange = function() {

if (xmlhttp.readyState == 4) {
```



```
alert ("file was last-modified-"+ xmlhttp.getResponseHeader("Last-Modified"));}
```

```
}
```

```
</script>
```

AJAX - getAllResponseHeader() method

The ajax - getAllResponseHeader() method returns the complete list of response headers.

this method has no arguments. note :- Each name/value pair is delimited by a carriage return/line feed (CR/LF) sequence.

ajax - getAllResponseHeader() method syntax:

```
varreturn_value = object.getAllResponseHeader();
```

ajax – getAllResponseHeader() method example

```
getAllResponseHeader()
```

```
<script>
```

```
varxmlhttp = new XMLHttpRequest();
```

```
xmlhttp.open("GET","./demofile.txt", true);
```

```
xmlhttp.send();
```

```
xmlhttp.onreadystatechange = function() {
```

```
if (xmlhttp.readyState == 4) {
```

```
alert (xmlhttp.getAllResponseHeader());
```

```
}}}
```

```
</script>
```




7.6 .5 AJAX Security Concerns

Ajax applications can be attacked in many ways, the following are common attacks:

i) XMLHttpRequest Vulnerabilities

AJAX uses the XMLHttpRequest(XHR) object for all communication with a server-side application, frequently a web service. A client sends a request to a specific URL on the same server as the original page and can receive any kind of reply from the server. These replies are often snippets of HTML, but can also be XML, Javascript Object Notation (JSON), image data, or anything else that Javascript can process.

Secondly, in the case of accessing an AJAX page on a non-SSL connection, the subsequent XMLHttpRequest calls are also not SSL encrypted. Hence, the login data is traversing the wire in clear text. Using secure HTTPS/SSL channels, which the modern day browsers support, is the easiest way to prevent such attacks from happening.

XMLHttpRequest(XHR) objects retrieve the information of all the servers on the web. This could lead to various other attacks such as SQL Injection, Cross Site Scripting (XSS), etc.

ii) Increased Attack Surface

Unlike traditional web applications that execute completely on the server, AJAX applications extend across the client and server, which gives the client some power. This throws in additional ways to potentially inject malicious content.

iii) SQL Injection

SQL Injection attacks (see Testing for SQL Injection) are remote attacks on the database in which the attacker modifies SQL statements before they are processed by the DBMS. Typical SQL Injection attacks could be as follows (examples refer to Microsoft SQL Server)



Example 1

```
SELECT id FROM users WHERE name=" OR 1=1 AND pass=" OR 1=1  
LIMIT 1;
```

This query will always return one row (unless the table is empty), and it is likely to be the first entry in the table. For many applications, that entry is the administrative login - the one with the most privileges. Note. The code fragment above tries to match userid and password values (obtained in input) with attributes name, pass of users; consequently, it appears that users is storing passwords in clear text, a practice which is not recommendable.

Video Lecture

<https://tinyurl.com/lvbdkzt>





Activity



Activity 6.0

Aim: Using XMLHttpRequest, Object to retrieve data

Motivation: To become conversant with Ajax technologies, and how to create Objects

Resources: Unit 6 notes, and examples

What to do:

After creating the object, we can send information to the web server.

Write proper Ajax Object's function for:

- i. opening connection to the server (use Post Method and it should process asynchronously)
- ii. send the data "`date=11-11-2006&name=Ali`".

How do we get the returned value? Write a property which retrieve information from the server

How to do it:

Read unit six notes, attain the basics on how to create objects then start working on this activity

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity for assessment to the course instructor and get a feedback on this activity.



Unit summary



In this Unit students learnt standards-based presentation using XHTML and CSS; dynamic display and interaction using the Document Object Model; data interchange and manipulation using XML and XSLT; asynchronous data retrieval using XMLHttpRequest. They learnt how Ajax technologies can be used to send data to, and retrieve data from, a server asynchronously without interfering with the display and behavior of the existing page.

Review Questions



1. Discuss disadvantages of Ajax
2. Explain what Ajax is.
3. Use skeleton example to explain how it is implemented
4. Discuss the technologies that are combined in Ajax
5. Explain how asynchronous processing is handled using Ajax

Reference and Further Reading



1. Testing for AJAX Vulnerabilities (OWASP-AJ-001). (n.d.). Retrieved October 04, 2016, from [https://www.owasp.org/index.php/Testing_for_AJAX_Vulnerabilities_\(OWASP-AJ-001\)](https://www.owasp.org/index.php/Testing_for_AJAX_Vulnerabilities_(OWASP-AJ-001))
2. Introduction to Ajax. (n.d.). Retrieved October 04, 2016, from <http://archive.oreilly.com/oreillyschool/courses/javascript2/IntroToAjax.html>



Attribution

This unit of *Introduction to AJAX* is a derivative copy of materials from [XML HttpRequest](#), licensed under [Creative Commons Attribution-ShareAlike License](#)

The following material was written by Adrienne Watt:

1. Notes
2. Activities



Unit 7

Introduction to Perl

Introduction

For a long time Perl has been a popular language among those programming for the first time. Although it is a powerful language many of its features mean make it especially suited to first time programmers as it reduces the complexity found in many other languages. Perl is also one of the world's most popular languages which mean there are a huge number of resources available to anyone setting out to learn it.

In this Unit you will learn how to write Perl program, variables, assigning values to scalars, features of Perl. You will learn different ways to run Perl program using different functions.

The course tries to provide grounding in the basic theory needed to write programs in any language as well as an appreciation for the right way to do things in Perl.

Unit outcomes



Upon completion of this unit you should be able to:

- Explain all features of Perl
- Describe different ways to run Perl program
- Understand Arithmetic in Perl
- Categorize different subroutines in Perl
- Explain the major difference between Perl and PHP



Terminologies



GUI	Graphical user interface (GUI), is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces
Error	An 'error' is a deviation from accuracy or correctness
UNIX	UNIX is a computer Operating System which is capable of handling activities from multiple users at the same time.
cd	Command used to change directory

7.1 Perl Background

7.1.1 History of Perl

Perl was introduced in 1987 (4 years before Linux itself), when the author, **Larry Wall**, released version 1.000 of it. The reason for its creation was that Wall was unhappy by the functionality that sed, C, awk and the Bourne Shell offered him. He looked for a language that will combine all of their best features, while having as few disadvantages of its own.

Since then, perl has seen several versions, each adding additional functionality. perl version 5, which was released in 1994, was a complete re-write of the perl interpreter, and introduced such things as hard references, modules, objects and lexical scoping. Several second-digit versions of perl 5 appeared since then, and the most up-to-date stable version (as of November 2010) is 5.12.x.

Perl became especially popular as a language for writing server-side scripts for web-servers. But that's not the only use of perl, as it is commonly used for system administration tasks, managing database data, as well as writing GUI applications.



7.1.2 Perl Development Life Cycle

Perl is interpreted, so no compilation is needed. To use perl, one should create a text file that contains the Perl program. This file can be written using any text editor available on your system. It is recommended that you end the filenames of your perl scripts with ".pl" in order to designate them as perl scripts.

After you are done, you should invoke the perl interpreter with the name of the file you created. Assuming your file is name "myscript.pl", you should type:

```
$ perl myscript.pl
```

At the command line (and press Enter). If your program contains errors or warnings they will be displayed. If your program does not contain errors, it will be executed. The fact that the program was executed does not mean it does what you want it to do. We will learn how to debug Perl programs, later on in this series.

7.1.3 Creating and Running Perl simple script

It is traditional when learning a new programming language to create a first program named *helloWorld*. This first program simply prints the words Hello, world! to the screen and quits. Once you are able to accomplish this, you have learned all the prior steps required for writing and running programs in the new programming language. Our script, which we will call helloWorld1.pl, is given below:

```
#!/usr/bin/perl  
  
print( "Hello, world!\n" );
```




You can either copy this code from this page and paste your editor's window, or type it in yourself. Save the file as `helloWorld1.pl`. Then from the command line, move to the directory in which you placed the `helloWorld1.pl` file, and enter the command:

```
perl helloWorld.pl
```

This command starts Perl, which then runs the script. If errors appear in your code, Perl will tell you where the errors are and quit before running the script. If your script looks fine, Perl will run it. The result should be that the words `Hello, world!` appear on the screen.

```
> perl helloWorld.pl
Hello, world!
```

The first line of the script is called the '*shebang*' line. The word *shebang* comes from the names of the first two characters of the line, the sharp (#) and the bang (!). These characters must be the first two characters of the file, and they are followed immediately by the pathname of the UNIX program that will run the script. In our example, we provide the pathname to Perl.

On a Macintosh computer running Mac OS X 10.3.4, the full path to Perl is `/usr/bin/perl`, so that is the path we'll use in all the examples of this tutorial.

7.1.4 Comments

In a Perl script, anything appearing after a '#' on a line is a comment. Comments are ignored by the Perl interpreter and thus provide a good way to provide information about a script. For every script you create, you should add comment lines that give:

1. the name of the script
2. the date the script was written
3. the name of the author of the script
4. contact information for the author
5. a brief description of what the script does



6. an example of how to run the script

Here's our helloWorld.pl script with a full set of comments:

```
#!/usr/bin/perl
#This file name is hellowrld2.pl
#
#Created on 13-Jun-2016
#
#Mathias Ombeni, Open University of Tanzania
#Contact:hod.ict@out.ac.tz
#
#This script prints "Hello, world!" to the
#screen and then quits.
#
# Use: perl helloWorld2.pl

print( "Hello, world!\n" );
```

By tradition, the first program you should write when you're learning a new language is one which prints the words "Hello World" on the screen, and then exits. It's surprising how much you can learn about a language just from being able to do this.

Our hello world script is called hello_world.pl and is shown below. Perl programs don't have to be named with a .pl extension but you will need to name them like this for windows to recognize that they're Perl scripts. It's also useful to keep this convention just so you can tell what your files are just by looking at the name.

In the script below I've added line numbers at the start of each line. These aren't part of the program, they're just there so I can refer back to them later on.

```
1 #!c:/perl/bin/perl.exe
2 use warnings;
3 use strict;
4 use diagnostics;
```



```
5
6 # A quick test script...
7
8 print "Hello World!\n";
```

To run this script use the “cd” command in your command shell to move to the directory where you created it, and then type:

```
perl hello_world.pl
```

You should see the words “Hello World!” printed to the screen before your command prompt returns to allow you to enter more commands.

7.2 How Perl Works

Now you’ve seen a working Perl script, let’s go through it so we can see how it works.

The first line in a Perl script should always be a pointer to the location of the perl interpreter you’d like to use to run the script. This is mostly only used on unix-like systems, but it’s good practice to include it even on windows-based scripts. The syntax for the line is #!(pronounced “hash – bang), followed by the path to perl.

From this point on your program is just a set of Perl statements. Each statement is usually put on a separate line, but is always terminated by a semi-colon. Perl doesn’t care how your code is laid out – it could all be on one line as far as it’s concerned, but it will make your code much more readable if it’s organized sensibly.

Unless instructed otherwise the perl interpreter will start at the top of your file and keep executing statements until it gets to the bottom, at which point it will exit.

Lines 2 - 4 tell the program that you’d like to introduce some extra functionality into your program from an external file called a Perl Module. Modules are a way of easily being able to extend the base Perl



language, and we'll talk a lot more about them later. For now, all you need to know is that:

Lines 2 and 3 are the Perl equivalent of fastening your safety belt. Perl, by default lets you get away with some really bad programming practices. This is because many people use Perl as a command line tool where they write a whole program in a single command, and to do this you need to save as much space as possible. The programs we'll be writing though aren't constrained by space and will therefore not cut corners and will be done properly!

Line 4 is useful when you're starting out with perl and can be omitted later on once you've been using it for a while. The effect of including this line is that if your program encounters an error you would usually just get a terse message pointing out what went wrong. By including the diagnostics module you also get a longer more friendly explanation of what this might mean. [On some macs we've found that 'use diagnostics' doesn't work unless you have the mac developer tools installed so if you get an error about this line just comment it out until you can install these]

Line 6 is a comment. If you put a hash (#) into a Perl script then everything from that point on up to the end of the line is ignored by the perl interpreter. Perl does not have separate syntax for multi-line comments. It's generally a good idea to include comments in your code to help explain the reasoning around a particular piece of code.

Line 8 is where the work actually happens. It uses the print function to print the text "HelloWorld!" to the screen. The "\n" at the end of the text indicates that perl should print a new line character (equivalent to pressing return).

7.3 Scalars and Scalar variables

The first thing we're going to look at in Perl is how it stores and manipulates data. In our hello world script we've actually already used some data – the string "Hello World!\n". If we'd changed that data then our program would have printed something different.



If we had some data we wanted to use in several places in our program, rather than typing it out each time we can store it in a variable. A variable is simply a way of associating some data with a short name which you can use to refer back to it later.

The Perl data structure which is used to hold a single item of data (such as a piece of text, or a number) is called a scalar. A variable which can store a piece of scalar data is called a scalar variable.

Scalar variables in Perl have names which start with a dollar sign, and then have a name which consists of letters, numbers and the underscore character. By convention they are usually put in all lowercase. Examples of typical variable names would be;

```
$x  
$name  
$first_name
```

Unlike a lot of other languages, Perl does not have a separate data type to hold characters, strings, integers and floating point numbers. The scalar variable type can hold all of these and perl will automatically convert them to the right kind of data depending on the context in which you use it (but as long as you have your safety belt fastened it will warn you if you try to do something stupid like “hello world”+3!).

7.3.1 Assigning values to scalars

To create a new scalar variable you use the syntax shown below;

```
my $first_name = "Simon";
```

When you want to create a new variable you need to use the keyword “my” in front of the variable name. This tells the parser that you know that you’re creating a new variable, and allows it to catch problems which occur from spelling mistakes such as the one below;

```
my $first_name = 'Simon';  
$frist_name = 'Bob';
```



If you tried to run this code you'd get the error shown below;

Global symbol "\$frist_name" requires explicit package name line 7.

Execution aborted due to compilation errors.

Declaring a new variable also sets up the variable's 'scope', that is it defines which parts of the program can see the variable you have created

7.3.2 Quoting

Data contained in single quotes is interpreted literally. Whatever characters you put between the quotes go into your variable.

```
my $var = 'This is some $text';  
print $var;
```

This would produce - This is some \$text – on the command line when run

If you use double quotes instead however, then certain parts of what you quote will be substituted for something else. The data in double quotes are said to be "interpolated".

There are two kinds of substitution which happen in double quotes, variable interpolation and the expansion of special characters. Below you can see an example of variable interpolation.

```
my $name = 'Simon';  
my $message = "Hello, my name is $name";  
print $message;
```

In this case what you would see printed is - Hello, my name is Simon. By using double quotes the \$name in the message will be substituted with the data contained in the \$name variable. Of course the example above shows and unnecessarily long way of doing this, and we could just do the interpolation in the print statement.

```
my $name = 'Simon';  
print "Hello, my name is $name";
```



Special characters are those which you might want to include in data, but which you can't type on a keyboard without doing other strange things to your program. The two most used special characters are the tab and the newline characters.

Special characters in Perl are single letters preceded by a backslash. The example below shows the use of both a tab and a newline.

```
print "1\t2\t3\nA\tB\tC\n";
```

This produces the output shown below, where the "\t" has been substituted for a tab, and the "\n" has become a newline.

```
1      2      3
A      B      C
```

The list of special characters is:

Character	Meaning
\a	Alarm (print a beep!)
\b	Backspace (lets you overwrite existing text)
\f	Form feed (move down but not back)
\n	New line (move down and back)
\r	Carriage Return (move back but not down)
\t	Tab

7.4 Perl Functions

Functions are the main way to perform an operation in perl. They are simply named blocks of code which perform a specific operation. Later in the course we will see that we can construct our own functions by writing sub-routines, but for now we're going to focus on built-in functions. Using a function in perl is achieved by using the construct shown below

```
my $result = function_name($data)
```



You simply use the name of the function to run it. Many functions rely on being provided with some data to work with so the way to provide this is by including a set of round brackets after the function name and then including in there the data the function needs to work with. If the function needs more than one piece of data you separate the different data pieces using commas. In many cases you can omit the brackets and just call the function as

```
my $result = function_name $data
```

You will often see this in scripts, but it's not generally a good idea, especially when you're first starting to write your own scripts. Keeping the brackets makes it very clear which data is going into the function, but to some extent it's a matter of style whether you include them or not.

All functions return some data after they have run. For some functions this data is empty or not useful, but in many cases you do want to keep the data passed back from the function.

To keep data from a function you can assign it to a variable the same way you would with a raw piece of data.

There are a number of perl built-in functions which you can begin to use once you have some scalar data.

7.4.1 Print

This is the one function you've seen so far. Print takes either a single scalar or a list (multiple scalars separated by commas) and by default prints them to STDOUT (the standard output –usually the console)

```
print "This is some text";  
  
print "I can however", "print more than one scalar", "in the  
same statement";
```

7.4.2 Length

The length function returns the length of the scalar

```
my $length = length("abcde");  
  
print $length; # prints 5
```




7.4.3uc / lc

The functions `uc` and `lc` can be used to convert a string into upper or lower case. They do not alter the original string, but instead return the adjusted string, which can be assigned to a new variable, or back to the original one.

```
my $mixed = "cASe is ALL oVeR The PlaCE";  
printlc($mixed); # All lower case, but $mixed unchanged  
$mixed = uc($mixed);  
print $mixed; # All upper case
```

7.4.4 Reverse

The reverse function reverses a scalar. As with `uc/lc` it doesn't change the scalar itself but returns a reversed version of it for you to play with.

```
my $string = "\n?sihtdaeruoynac";  
my $reversed = reverse $string;  
print $reversed;
```

7.4.5 Substr

The `substr` function allows you to extract a substring from a string. Its syntax is

```
substr ([string],[offset],[length])
```

String is the scalar you want to extract the substring from Offset is the position in the string you want to start from (counting from 0). If you want to be clever you can use a negative offset to start counting back from the end of the string. Length is the length of substring you want to extract.



Video Lecture

<https://tinyurl.com/ke4mxns>





Activity



Activity 7.0

Aim: Write Perl program to read file

Motivation: To become conversant with different ways of manipulating information from file using Perl program

Resources: Unit 7 notes

Tools: Computer and a text editor such as notepad++

What to do:

Write a program to read a file from disk, and save it as a new text file. Try downloading any public-domain book and test how your program works with large text files

How to do it:

Read notes and examples given in unit 7 and links provided then follow instructions on how to do this activity from similar activities provided on the links.

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity to course instructor for assessment and feedback.



Unit summary



This unit has covered basic concepts on how to write Perl program, how to declare variables in Perl, assigning values to scalars and features of Perl. They learnt different ways to run Perl program using different functions.

The grounding in the basic theory will assist in writing programs in any programming language as well as an appreciation for the right way to do things in Perl.

Review questions



1. There are many types of primary data structure in Perl. List them and explain what they mean
2. There are many types of operators, but how many types of operators are used in the Perl
3. In Perl we can show the warnings using some options in order to reduce or avoid the errors. Outline the options.

Reference and Further Reading



1. Wikibooks, Perl Programming/Functions. (n.d). Retrieved September 14, 2016, from https://en.wikibooks.org/wiki/Perl_Programming/Functions



Attribution

This unit of *Introduction to Perl* (including images, except as otherwise noted) is a derivative copy of materials from [Learning Perl](#) licensed under [Creative Commons Attribution License 3.0 license](#)

The following material was written by Andrews, S:

1. Notes



Unit 8

Introduction to jQuery

Introduction

The jQuery library makes it easy to manipulate a page of HTML after it's displayed by the browser. It also provides tools that help a user to interact with a page, tools that help to create animations in a page, and tools that let user communicate with a server without reloading the page.

In this lecture students will learn some jQuery basics, and how jQuery can be used to perform its core functionality: getting some elements and doing something with them.

Unit outcomes

Upon completion of this unit you should be able to:



- Understand how to create and use JQuery
- Explain all feature of jQuery
- Explain how jQuery Works
- Explain when to use jQuery
- Understand useful jQuery Functions
- Understand how to read, write and delete cookies in jQuery

Terminologies



jQuery:	It provides interactions, widgets, and effects for creating Rich Internet Applications.
Events:	Methods trigger or attach a function to an event handler for the selected elements
Effects:	Techniques for adding animation to a web page elements
Selector:	Allow you to select and manipulate HTML element(s).



8.1 Understanding jQuery

8.1.1 What is JQuery

The jQuery syntax is used for selecting HTML elements and performing some action on the element(s). Note: - the jQuery doesn't change original HTML and CSS files. It makes changes only to the DOM representation of the page in browser memory.

Basic syntax is :- `$(selector).action()` or `jQuery(selector).action()`

The dollar sign with the parenthesis is the shorter name of the jQuery function.

- \$ sign to define/access jQuery
- a (selector) to hold HTML dom elements
- a jQuery action() to be performed on the element(s)

Examples

`$("p").hide()` :- hides the all paragraph elements.

`$("p").remove()` :- removes the all paragraph elements.

`$(".date").hide()` :- hides the all elements with class="date".

`$("#date").remove()` :- removes the all elements with id="date".

`$(this).hide()` :- hides the current element.

8.1.2 What Can jQuery Do

DOM traversal and manipulation

Get the <button> element with the class 'continue' and change its HTML to 'next step...'

```
$( "button.continue").html( "Next Step--")
```



8.1.3 jQuery - Event Handling

Show the #banner element that is hidden with display:none in its CSS when any button in #button-container is clicked.

```
var hiddenbox = $("#banner");  
    $("#button-container button").on("click",function( event ) {  
hiddenbox.show();  
    });
```

8.1.4 Downloading jQuery

There are two versions of jQuery available for downloading:

- Development version -this is for testing and development.
- Production version -this is for your live website (it is minified and compressed.)

jQuery versions adding to your website follow two simple steps.

- Go to the download page to grab the latest version available.
- Now put downloaded jquery-1.9.1.js file in a directory of your website (e.g /jquery//)

Now you can include jQuery library in your HTML file as follows:--

Include jQuery

```
<html>  
<head>  
<title> how you can include jQuery library in your html  
file. </title>  
<script type="text/javascript" src="/jquery/jquery-1.9.3.js"  
>  
</script>  
</head>  
</html>
```

Using CDN(Content Delivery Network)

If you don't want to download and host jQuery yourself, then you can include it from a CDN (Content Delivery Network).



Advantages

- Decrease server load.
- Faster content delivery.
- 100% availability.

To use jQuery from Google or Microsoft, use one of the following:--

Google cdn

```
<html>
<head>
<script
src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js" >
</script>
</head>
</html>
```

Microsoft cdn

```
<html>
<head>
<script src="//ajax.aspnetcdn.com/ajax/jquery/jquery-1.9.1.min.js"
>
</script>
</head>
</html>
```

8.2 jQuery Selector

jQuery selectors select elements to add behaviour to those elements.

jQuery selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria.

jQuery selector is a powerful set of tools for matching a set of elements in a document.

All type of selectors in jQuery, start with the dollar sign and parentheses: \$(selector).



8.2.1 The Universal Selector

The `*` (universal) selector selects all elements in the document, including `html`, `head`, `body`. If the `*` (universal) selector is used together with another element, it selects all child elements within the specified element. The syntax:-

```
$("#*") or jQuery("*")
```

Example

`("*")` selector

```
<script>
```

```
$(document).ready(function(){  
    $("body>*").css("background-color","orange");  
}); </script>
```

8.2.2 The Element Selector

The jQuery element selector is used to select elements based on their tag names. JQuery can now restrict the search to `DIV` element only. The syntax:-

```
$("#element") or jQuery("element")
```

Example

`("*")` selector

```
<script>
```

```
$(document).ready(function(){  
    $("li").css({"background-color":"yellow","border":"1px dotted pink"});  
});
```

```
</script>
```

8.2.3 The #Id Selector

The jQuery `#id` selector uses the `id` attribute of an HTML tag to find the specific element. Tip:- HTML `#ID` attributes are unique in every page and even all browsers can locate a single element very quickly. The syntax:-

```
$("#id") or jQuery("#id")
```



Example

```
("#id) selector

<script>

$(document).ready(function(){

$("#head").css({"background-color":"#4a3b9c","color":"#ffffff"});

$("#disc").css({"background-color":"yellow","border":"1px dotted pink"});

});

</script>
```

8.2.3 The Class Selector

The jQuery class selector finds element with a specific class. The class selector will run quickly in modern browsers. The syntax:-

```
$(".class") or jQuery(".class")
```

The class selector will be more efficient if we qualify it with a tag name e.g. \$("div.myclass");

Example

(".class") selector

```
$(document).ready(function(){

$(".head").css({"background-color":"#4a3b9c","color":"#ffffff"});

$(".disc").css({"background-color":"yellow","border":"1px dotted pink"});

}); </script>
```

[Click here for jQuery selectors](#)

8.3 jQuery Events

We have the ability to create dynamic web pages by using events. Events are actions that can be detected by your web application.

Examples:-

- A mouse click
- Moving a mouse over an element.



- Clicking on an element.
- A web page loading.
- A keystroke on your keyboard

Activity



Activity 8.0

jQuery basic hands-on activities

Aim: Describe how jQuery works with CSS and HTML.

Motivation: To become conversant in using jQuery in connection with CSS to improve interactivity to web application and websites

Resources: Unit 8 notes



Tools: Notepad ++ or any related text editor, JavaScript Library

What to do:

- Find and Apply background to all paragraph elements in HTML files.
- Set the background colour of a page to red.
- Find the specific option tag text value of a selected option

How to do it:

Read notes and examples given in unit 8 and link provided then follow instructions on how to do similar activities provided on the links.

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity to course instructor for assessment and feedback.

Activity 8.1

Aim: Demonstrate how jQuery Events and Effects can be added to HTML element.

Motivation: To become conversant in adding jQuery Events and Effects to HTML elements to improve interactivity to web application and websites.

Resources: Unit 8 notes

Tools: Notepad ++ or any related text editor, JavaScript Library

What to do:

Find all inputs that are descendants of a form and mark them with a dotted red border

Note: Descendant Selector ("ancestor descendant") selects all elements that are descendants of a given ancestor. A descendant of an element could be a child, grandchild, great-grandchild, and so on, of that element.

```
<body>
```

```
<form>
```



```
<label for="name">Child of form:</label>
<input name="name" id="name">
<fieldset>
<label for="newsletter">Grandchild of form, child of fieldset:</label>
<input name="newsletter" id="newsletter">
</fieldset>
</form>
Sibling to form: <input name="none">
</body>
```

Duration: Expect to spend about 30 min on this activity

Feedback: The learner should submit this activity to course instructor for assessment and feedback.

Unit summary



In this Unit, we have covered about jQuery. Students were equipped with basic JavaScript skills and knowledge that enable them to use jQuery as JavaScript Library to make improve to navigate animation and handle events to HTML a document. In this Unit students learnt some jQuery basics, and at how jQuery can be used to perform its core functionality: getting some elements and doing something with them.



Review Questions



1. Does jQuery use CSS selectors to select elements? If Yes How
2. Which sign does jQuery use as a shortcut for jQuery?
3. What does the following selector: \$("div") mean?
4. Is jQuery a library for client scripting or server scripting? Explain your answer
5. Is it possible to use jQuery together with AJAX?
6. Which jQuery method is used to hide selected elements
7. Explain the three types of jQuery selectors

Reference and Further reading



1. JQuery Basics. (2014). Retrieved September 14, 2016, from <http://jqfundamentals.com/chapter/jquery-basics>
2. Murphey, R. (2011). JQuery Fundamentals. Open Educational Resource. Retrieved September 14, 2016, from <http://acbc.com.au/admin/images/uploads/Copy4jQuery-Tutorial.pdf>

Attribution

This unit of *Understanding jQuery* is a derivative copy of materials from [jQuery Basics](#), licensed under [Creative Commons Attribution License 4.0 license](#)

The following material was adapted from the material:

1. Notes





Unit 9

Introduction to Common Gateway Interface (CGI)

Introduction

This Unit is about software interface between the Web server and programs. CGI is the part of the Web server that can communicate with other programs running on the server. With CGI, the Web server can call up a program, while passing user-specific data to the program (such as what host the user is connecting from, or input the user has supplied using HTML form syntax). The program then processes that data and the server passes the program's response back to the Web browser. In this Unit you will learn how to install CGI programs on the web servers, how CGI technology works and acquire understanding on environment variables passed to CGI programs.

Unit outcomes



Upon completion of this unit you should be able to:

- .Identify and understand CGI environment variables
- Explain what is CGI
- Understand and describe the uses for CGI
- Describe Drawbacks of using CGI
- Explain how CGI technology works.
- Install CGI programs on the web servers.
- Identify the ways used to connect client browsers to CGI Programs



Terminologies



Syntax	Syntax is the set of rules that defines the combinations of symbols that are considered to be a correctly structured document or fragment in that language.
Deployment	Deployment is the use of something or someone in an effective way
HTTP	The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.
Web server	A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients.

9.1 Understanding Common Gateway Interface (CGI)

Common Gateway Interface (CGI) is a standard way for web servers to interface with executable programs installed on a server that generate web pages dynamically. Such programs are known as CGI scripts or simply CGIs; they are usually written in a scripting language, but can be written in any programming language.

Uses

CGI is often used to process inputs information from the user and produce the appropriate output. An example of a CGI program is one implementing a Wiki. The user agent requests the name of an entry; the Web server executes the CGI; the CGI program retrieves the source of that entry's page (if one exists), transforms it into HTML, and prints the result. The web server receives the input from the CGI and transmits it to the user agent. If the "Edit this



page" link is clicked, the CGI populates an HTML text area or other editing control with the page's contents, and saves it back to the server when the user submits the form in it.

9.2 How CGI Works

Each Web server runs HTTP server software, which responds to requests from Web browsers. Generally, the HTTP server has a directory (folder), which is designated as a document collection — files that can be sent to Web browsers connected to this server. For example, if the Web server has the domain name `example.com`, and its document collection is stored at `/usr/local/apache/htdocs` in the local file system, then the Web server will respond to a request for `http://example.com/index.html` by sending to the browser the (pre-written) file `/usr/local/apache/htdocs/index.html`.

CGI extends this system by allowing the owner of the Web server to designate a directory within the document collection as containing executable scripts (or binary files) instead of pre-written pages; this is known as a CGI directory. For example, `/usr/local/apache/htdocs/cgi-bin` could be designated as a CGI directory on the Web server. If a Web browser requests a URL that points to a file within the CGI directory (e.g., `http://example.com/cgi-bin/printenv.pl`), then, instead of simply sending that file (`/usr/local/apache/htdocs/cgi-bin/printenv.pl`) to the Web browser, the HTTP server runs the specified script and passes the output of the script to the Web browser. That is, anything that the script sends to standard output is passed to the Web client instead of being shown on-screen in a terminal window.



Syntax

The following Perl program shows all the environment variables passed by the Web server:

```
#!/usr/bin/perl
=head1 DESCRIPTION
printenv — a CGI program that just prints its
environment=cut
print "Content-type: text/plain\n\n";

for my $var ( sort keys %ENV )
{
    printf "%s = \"%s\"\n", $var, $ENV{$var};
}
```

If a Web browser issues a request for the environment variables at <http://example.com/cgi>

[bin/printenv.pl/foob/bar?var1=value1&var2=with%20percent%20encoding,a64-bit](http://example.com/cgi-bin/printenv.pl/foob/bar?var1=value1&var2=with%20percent%20encoding,a64-bit) Microsoft Windows web server running cygwin returns the following information:

From the environment, it can be seen that the Web browser is Firefox running on a Windows 7 PC, the Web server is Apache running on a system that emulates Unix, and the CGI script is named `cgi-bin/printenv.pl`.

The program could then generate any content, write that to standard output, and the Web server will transmit it to the browser.

The following are environment variables passed to CGI programs:

From the environment, it can be seen that the Web browser is Firefox running on a Windows 7 PC, the Web server is Apache running on a system that emulates Unix, and the CGI script is named `cgi-bin/printenv.pl`.



The program could then generate any content, write that to standard output, and the Web server will transmit it to the browser.

The following are environment variables passed to CGI programs:

Server specific variables:

- `SERVER_SOFTWARE`: name/version of HTTP server.
- `SERVER_NAME`: host name of the server, may be dot-decimal IP address.
- `GATEWAY_INTERFACE`: CGI/version.

Request specific variables:

- `SERVER_PROTOCOL`: HTTP/version.
- `SERVER_PORT`: TCP port (decimal).
- `REQUEST_METHOD`: name of HTTP method (see above).
- `PATH_INFO`: path suffix, if appended to URL after program name and a slash.
- `PATH_TRANSLATED`: corresponding full path as supposed by server, if `PATH_INFO` is present.
- `SCRIPT_NAME`: relative path to the program, like `/cgi-bin/script.cgi`.
- `QUERY_STRING`: the part of URL after `?` character. The query string may be composed of `*name=value` pairs separated with ampersands (such as `var1=val1&var2=val2...`) when used to submit form data transferred via GET method as defined by HTML application/x-www-form-urlencoded.
- `REMOTE_HOST`: host name of the client, unset if server did not perform such lookup.
- `REMOTE_ADDR`: IP address of the client (dot-decimal).
- `AUTH_TYPE`: identification type, if applicable.
- `REMOTE_USER` used for certain `AUTH_TYPES`.



- `REMOTE_IDENT`: see `ident`, only if server performed such lookup.
- `CONTENT_TYPE`: Internet media type of input data if `PUT` or `POST` method are used, as provided via HTTP header.
- `CONTENT_LENGTH`: similarly, size of input data (decimal, in octets) if provided via HTTP header.
- Variables passed by user agent (`HTTP_ACCEPT`, `HTTP_ACCEPT_LANGUAGE`, `HTTP_USER_AGENT`, `HTTP_COOKIE` and possibly others) contain values of corresponding HTTP headers and therefore have the same sense.

The program returns the result to the Web server in the form of standard output, beginning with a header and a blank line.

The header is encoded in the same way as an HTTP header and must include the MIME type of the document returned.[8] The headers, supplemented by the Web server, are generally forwarded with the response back to the user.

Here is a simple CGI program in Python along with the HTML that handles a simple addition problem.[9]

```
<!DOCTYPE html>
<html>
<body>
<form action="add.cgi" method="POST">
  Enter two numbers to add:<br />
  First Number: <input type="text" name="num1" /><br />
  Second Number: <input type="text" name="num2" /><br />
  <input type="submit" value="Add" />
</form>
</body>
```



```
</html>

#!/usr/bin/python

importcgi
importcgitb

cgitb.enable()

input_data=cgi.FieldStorage()

print 'Content-Type:text/html' #HTML is following

print #Leave a blank line

print '<h1>Addition Results</h1>'

try:

    num1=int(input_data["num1"].value)

    num2=int(input_data["num2"].value)

except:

print '<p>Sorry, we cannot turn your inputs into numbers
(integers).</p>'

return 1

sum=num1+num2

print '<p>{0} + {1} = {2}</p>'.format(num1,num2,sum)
```

9.3 CGI Deployment

A Web server that supports CGI can be configured to interpret a URL that it serves as a reference to a CGI script. A common convention is to have a `cgi-bin/` directory at the base of the directory tree and treat all executable files within this directory (and no other, for security) as CGI scripts. Another popular convention is to use filename extensions; for instance, if CGI scripts are consistently given the extension `.cgi`, the web server can be configured to interpret all such files as CGI scripts. While



convenient, and required by many pre-packaged scripts, it opens the server to attack if a remote user can upload executable code with the proper extension.

In the case of HTTP PUT or POSTs, the user-submitted data are provided to the program via the standard input. The Web server creates a subset of the environment variables passed to it and adds details pertinent to the HTTP environment.

Video Lecture

<https://tinyurl.com/lafkr99>





Activity



Activity 9.0

Aim: Create an HTML form and their values to a CGI program

Motivation: To become conversant with CGI programming

Resources: Unit 9 notes

What to do:

HTML Form

Using the example of your own selection, create an HTML form in your public_html directory which will pass at least three identifiers of different input types (e.g., radio, checkbox, text, textarea) and their values to a CGI program.

You may pass as many values as you like, but pass at least three. Be certain to name your HTML forms file cgiactivity9.html.

How to do it:

Read notes and examples given in unit 9 and link provided then follow instructions on how to do similar activities provided on the links.

Duration: Expect to spend about 1 hour on this activity.

Feedback: This is a peer-reviewed assessment where learner's findings on this activity will be assessed and compared with other learner's findings on the same activity in class.



Unit summary



In this unit, students learnt how to install CGI programs on the web servers and how CGI technology works. Students also acquired knowledge and skills on how environment variable passes to CGI programs.

Review questions



1. In a CGI script, identify the CGI environment variable that contains the URL of the HTML form that invoked the CGI script
2. CGI script can maintain sessions without storing any information on
3. Where does the data for the CGI Script come from?
4. What are the Drawbacks of using CGI?
5. The CGI input information can be roughly broken into three groups. With examples write down the groups.

Reference and Further reading



1. Anthony, G. (2000). Common gateway interface (CGI). Retrieved September 14, 2016, from <http://www.peoi.org/Courses/Coursesen/web/web3.html>



Attribution

This unit of *Introduction to Common Gateway Interface (CGI)* is a derivative copy of materials from [Common Gateway Interface](#). Licensed under [Creative Commons Attribution-ShareAlike License](#)

The following material was adapted from the link:

1. Notes
2. Activities



Unit 10

Web Application Security

Introduction

This topic introduces Web application security, explains common security terminology and presents a set of security principles. It presents an overview of the security process and explains why a holistic approach to security that covers multiple layers including the network, host and application, is required to achieve the goal of hack-resilient Web applications.

This lecture will introduce and define host configuration categories and application vulnerability categories. It will also discuss security requirements in web tier and enterprise tier applications. You will learn solutions for different categories of threats in web-enabled security implementations. Finally, you will learn how to assess and apply modern website design principles in the area of web technology, Internet marketing, usability and accessibility.

Unit outcomes



Upon completion of this unit you should be able to:

- Describe how hackers find security vulnerabilities!
- Explain how hackers exploit web applications!
- Describe different types of web sites and web system's attacks.
- Describe Solution to web sites and web system's attacks.



Terminologies



Access Control	The selective restriction of access to web resource
Authentication	A process in which the credentials provided are compared to those on file in a database
Authorization	Specifying access rights to web resources
CMS	A computer application that supports the creation and modification of digital content using a common user interface
Module	A collection of code files that adds one or more features/function to your web application.
Plugin	A software component that adds a specific feature to an existing computer program

10.1 Understanding Web Application Security

Web application security is a branch of Information Security that deals specifically with security of websites, web applications and web services.

10.1.1 The primary web security controls

- **Confidentiality**

Refers to the protection of web information from unauthorized disclosure to a person or computing entity .

- **Access control (authentication and authorization)**

Authenticates the identity of the entity trying to access a website, web application and web resources, and controls the use of those resources per predetermined levels of entitlement. Authentication can be done by *static password*, *token*, *one time password*, *biometrics authentication*, *public* and *private key* authentication.

- **Integrity**



Controls and protect the web resources from any intentional or unintentional tampering. It ensures that web information or resources is not change by unauthorized person.

- **Availability**

Refers to the continuity of the availability of web resources. Adequate configuration of the system and controlled processes and procedures guards against denial of services attack.

10.1.2 Common words used in computer security

- **Threat**

A threat is an agent that may want to or definitely can result in harm to the target organization.

- **Vulnerability**

Vulnerability is some flaw in our environment that a malicious attacker could use to cause damage in your organization.

- **Risk**

Risk is where threat and vulnerability overlap. That is, we get a risk when our systems have a vulnerability that a given threat can attack.

- **Exploit**

An exploit is the way or tool by which an attacker uses a vulnerability to cause damage to the target system i.e website, web application and resources

- **Hacking**

is an act of modifies website or web application in a way that alters the creator's original intent. *Hackers* is a person who hack web application.



10.2 Main Types of Website Attacks

10.2.1 List of types of website's and web system's attacks.

- Outdated Programs exploitation
- SQL injection
- Shell script upload
- Cross Site Scripting (XSS)
- Cookie/Session hijacking
- Backdoor
- Brute Force/ dictionary attack

10.2.2 Common mistakes that introduces security holes in websites and web systems

- Using unknown application (CMS, Modules, templates etc.)
- Outdated CMS & Programs
- Poor programming (Java script validation, detailed error messages, simple escaping functions/NO sanitation, using super user in database's connection script, poor files & users permissions , poor verification system , etc.)
- Unprotected administrator directory
- Unrestricted search engine administrator directory crawling
- Unprotected directories access

10.2.3 How Hackers Detect existence of security holes in Websites or Web systems

There are mainly two types of penetration testing

- Without using Tools(*Require practical demonstrations*)
- Using Tools (SQL map, SQL ninja, Joomscan, *WPScan*, Firefox add-ons "*Firebug,SQL Inject Me, XSS Me*", BackTrack 5 "*Compiled tools*")



10.3 Detection and Solution of the Attacks

10.3.1 Categories of attacks

- Outdated Programs exploitation
- SQL injection
- Shell script upload
- Cross Site Scripting (XSS)
- Cookie/Session hijacking
- Backdoor
- Brute Force/ *dictionary attack*

10.3.2 Preventing websites and web applications from being hacked

- Avoid using unknown web application (CMS, Modules, etc.)
- Avoid outdated CMS & Programs
- Avoid Poor programming (Java script validation, detailed error messages, simple escaping functions/NO sanitation, using super user in database's connection script, poor files & users permissions, poor verification system, etc.)
- Avoid Unprotected administrator directory
- Avoid Search engine administrator directory crawling
- Avoid Unprotected directories access

10.3.3 Solution to website's and web system's attacks.

1) Outdated Programs exploitation

Mainly programs upgrades occur for three reasons:

- Adding new features
- Architecture Improvement (eg. execution speed improvement)
- Security updates

Solution: Avoid old versions

2) SQL Injection Attacks

SQL Injection attacks are attacks aimed at exploiting databases. This leads to Operating System attack and take over control. These



vulnerabilities can occur if user input is not filtered for escape characters (' " \ * % # ^ etc).

Solution: use function to filter escape characters

3) Shell script upload

This is uploading malicious script to web server and use the shell to exploit the whole server.

- Upload .exe file into web tree - victims download trojaned executable
- Upload virus infected file - victims' machines infected
- Upload .html, .php, .asp, .jsp, etc : file containing script - victim experiences

4) Cross-site Scripting (XSS)

Unprotected upload field can be used to this kind of attack. Poor protected upload file field can be used to upload shell using advanced skill.

Solution: All the control characters and Unicode ones should be removed from the filenames and their extensions without any exception. Also, the special characters such as “,”, “.”, “>”, “<”, “/”, “\”, additional “.”, “*”, “%”, “\$”, and so on should be discarded as well.

5) Cross Site Scripting (XSS)

Cross Site Scripting also known as XSS, is one of the most common web application

Vulnerability that allows an attacker to run his own client side scripts (especially JavaScript) into web pages viewed by other users.

Solution: sanitation

6) Cookie/Session hijacking

The attacker uses a sniffer to capture a valid token session called “Session ID”, then he

uses the valid token session to gain unauthorized access to the Web Server without User-name or password.

7) Backdoor

The un-trusted script programmer may sometimes allow a back door so that the program can be accessed without user permission.



Solution: Avoid deliberate backdoor access or using programs from unknowing source

8) Brute Force/ dictionary attack

A dictionary attack uses a targeted technique of successively trying all the words in an exhaustive list called a dictionary (from a pre-arranged list of values)

Solution: Limiting number of login with respect to time.

Activity



Activity 10.0

Aim: Describe Solution to web sites and web system's attacks.

Motivation: Capability to search identify and use the right methods and devices for testing an application for vulnerabilities

Resources:

- Unit 10 learning materials
- Hyperlinks given in unit 10 notes

What to do:

There are a number of ways of testing an application for vulnerabilities such as SQL Injection.

- i. First search and list methods available for testing SQL injection vulnerabilities
- ii. Second use one of the methods in (i) to elaborate how you can test an application for vulnerability

Duration: Expect to spend about 1 hour on this activity

Feedback: The learner should submit this activity to course instructor for assessment and feedback.



Unit summary



This unit introduced host configuration categories and application vulnerability categories. It also discussed security requirements in web tier and enterprise tier applications. You learnt solutions for different categories of threats in web-enabled security implementations. You also learnt how to assess and apply modern website design principles in the area of web technology, Internet marketing, usability and accessibility.

Review questions



1. Describe the term Web Security.
2. Define the term *Confidentiality*, *Integrity* and *Availability* as applied in Web Security.
3. Define Threat, Vulnerability, Risk and Exploit.
4. Describe Computer Hacking.
5. Why do we need Ethical Hacking?
6. List and describe types of web system's attacks.
7. What are the Common mistakes that introduce security holes in websites and web systems?
8. Describe the solutions to common mistakes



Reference and Further reading



1. SQL SQL Injection. (2016, April 10). Retrieved September 14, 2016, from https://www.owasp.org/index.php/SQL_Injection
2. Category:Attack. (n.d.). Retrieved September 7, 2016, from <https://www.owasp.org/index.php/Category:Attack>

Attribution

This unit of *Web Application Security* is a derivative copy of materials from [Web application security](#) licensed under [Creative Commons Attribution-Share Alike License](#)

The following material was adapted from the link:

1. Notes
2. Activities

Video 20 – Review Questions and Activities

<https://tinyurl.com/k3qpugn>

