# Software Learning

## GitHub

GitHub is a code hosting platform for version control and collaboration. It lets us work together on projects from anywhere. Create a GitHub account first.

### STEP 1: Create a Repository

It is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheet, and data-sets anything that our project needs

* In the upper right corner, next to your avatar or identicon, click + and then select New repository.
* Name your repository hello-world.
* Write a short description.
* Select Initialize this repository with a README.



Fig 1. GitHub, Creating New Repository

### STEP 2: Branching

It is the way to work on different versions of a repository at one time. By default, our repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master. When we create a branch off the master branch, you’re making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while we were working on your branch, we could pull in those updates.



Fig 2. GitHub, Branching

#### To create a new branch

Go to your new repository hello-world.

Click the drop down at the top of the file list that says branch: master.

Type a branch name, readme-edits, into the new branch text box.

Select the blue Create branch box or hit “Enter” on your keyboard.

Now we have two branches, master and readme-edits. They look exactly the same. Next, we’ll add our changes to the new branch.

### STEP 3: Commit Changes

We are on the code view for our readme-edits branch, which is a copy of master.

On GitHub, **saved changes** are called **commits**. Each commit has an associated commit message, which is a description explaining why a particular change was made. Commit messages capture the history of our changes, so other contributors can understand what we have done and why.

### STEP 4: Open a Pull Request

When we open a pull request, we are proposing our changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

As soon as we make a commit, we can open a pull request and start a discussion, even before the code is finished. By using GitHub’s **@mention** system in your pull request message, you can ask for feedback from specific people or teams. We can even open pull requests in your own repository and merge them ourselves.



Fig 3. GitHub, Open a Pull Request

### STEP 5: Merge our Pull Request

In this final step, it’s time to bring our changes together – merging our readme-edits branch into the master branch.

Click the green Merge pull request button to merge the changes into master.

Click Confirm merge.

Go ahead and delete the branch, since its changes have been incorporated, with the Delete branch button in the purple box.



Fig 4. GitHub, Merge the Pull Request

## Arduino

Download the repository

[**https://github.com/geetarista/ST2-Arduino**](https://github.com/geetarista/ST2-Arduino)

Unzip the file and then open the sublime editor and copy paste the folder in Sublime Editor 🡪 Preferences 🡪 Browse Packages 🡪 (Paste it over here)

You need to have package control installed in sublime!

Get the package control code the web site and copy & paste code from the web to the editor **(CTRL + `)**

Then press **(CTRL + Shift + P)** and type in **Package Control: Install Package.** Type in **arduino** then download it.

Sublime will download some additional files after it. Then restart the sublime editor and start typing the code and upload the code to your microcontroller before you do keep in mind about the port

## Batch Command

|  |  |
| --- | --- |
| ECHO | Displays text on the screen |
| @ECHO OFF | Hides the text that is normally output |
| START | Run a file with its default application |
| REM | Inserts a comment line in the program |
| MKDIR/RMDIR | Create and remove directories |
| DEL | Deletes a file or files |
| COPY | Copy a file or files |
| XCOPY | Allows you to copy files with extra options |
| FOR/IN/DO | This command lets you specify files. |
| TITLE | Edit the title of the window. |

## Configuration (INI)

|  |  |
| --- | --- |
| [ ] Square Bracket means Comment | [Database] |
| user is the variable, and the value is root | user = root |
| Pass is the variable, and the value is empty string | pass = '’ |

## Hyper Text Access

It is important to remember that an .htaccess file will affect the directory it is placed in and all resulting sub-directories. Therefore, if you add your ‘.htaccess’ file to the ‘web site root’ then it will affect all subsequent folders like so:

http://www.yourdomain.com/

| -- directory1

| -- directory2

| -- directory3

| | -- directory3/childdirectory1

| | -- directory3/childdirectory2

| -- .htaccess

| -- index.html

However, if you place the ‘.htaccess’ file in http://www.yourdomain.com/directory1 then the features of the ‘.htaccess’ will be restricted to that folder and all child folders only. For example:

http://www.yourdomain.com/

| -- directory1

| | -- directory1/childdirectory1

| | -- directory1/childdirectory2

| | -- directory1/childdirectory3

| | | -- directory1/childdirectory3/newdirectory1

| | | -- directory1/childdirectory3/newdirectory2

| | -- .htaccess

| | -- index.html

After editing your .htaccess file on multiple occassions it may look a little complicated so I would recommend implementing comments. To do this, simply place the hash symbol at the beginning of every line like so:

**Comment**

# another comment here

**Directory Index**

You can change a default index file of directory with:

DirectoryIndex welcome.html welcome.php

**Custom Error Pages**

You can redirect your users to an error page with:

ErrorDocument 404 error.html

ErrorDocument 400 /400.html

ErrorDocument 401 /401.html

ErrorDocument 403 /403.html

ErrorDocument 404 /404.html

ErrorDocument 500 /500.html

ErrorDocument 502 /502.html

ErrorDocument 504 /504.html

But remember to create your error pages!

**Remove ‘www’**

RewriteEngine On

RewriteBase /

RewriteCond %{HTTP\_HOST} ^www.yourdomain.com [NC]

RewriteRule ^(.\*)$ http://yourdomain.com/$1 [L,R=301]

**Force “File Save As”**

If you would like force users to download files rather than view them in the browser you could use:

AddType application/octet-stream .csv

AddType application/octet-stream .xls

AddType application/octet-stream .doc

AddType application/octet-stream .avi

AddType application/octet-stream .mpg

AddType application/octet-stream .mov

AddType application/octet-stream .pdf

AddType application/octet-stream .avi .mpg .mov .pdf .xls .mp4

**Rewrite URLs**

If you would like to make your URLs a little easier to read (ie changing content.php?id=92 to content-92.html) you could implement the following ‘rewrite’ rules:

RewriteEngine on

RewriteRule ^content-([0-9]+).html$ content.php?id=$1

Redirect Browser to https

This is always useful for those who have just installed an SSL certificate:

RewriteEngine On

RewriteCond %{HTTPS} !on

RewriteRule (.\*) https://%{HTTP\_HOST}%{REQUEST\_URI}

**Simple Example**

RewriteEngine on

## Manually Routing All the Frontend php files ##

### RewriteRule ^/?$ Frontend/Index.php [NC,L] ###

RewriteRule ^/?$ Frontend/Home.php [NC,L]

RewriteRule ^apartments/?$ Frontend/Apartments.php [NC,L]

## Java

For JDK version jdk1.7.0\_07 Just update this with location of your JDK when a new version is released.

* Click START
* Type "Path" (without the quotes) into the search area
* You should see "Edit environment variables for your account" <--- click this
* A window should appear titled "Environment Variables"
* Click TEMP on the top area
* Scroll a little bit on the bottom second area until you find Path
* Select Path and click Edit...
* Paste this in at the very end of bottom text area
* ;C:\Program Files\Java\jdk1.7.0\_07\bin
* Make sure to OK out of both windows
* Restart Sublime text if needed

Get compiling and running your Java programs after completing the above.

**Use this in Sublime Editor**

Tools 🡪 Build System 🡪 New Build System 🡪 Create a new file JavaC-build 🡪 then copy & paste the code

{

"shell\_cmd": "javac -Xlint \"${file}\"",

"file\_regex": "^(...\*?):([0-9]\*):?([0-9]\*)",

"working\_dir": "${file\_path}",

"selector": "source.java",

"variants":

[

{

"name": "Run",

"shell\_cmd": "java \"${file\_base\_name}\""

}

]

}

Ctrl + Shift + B choose JavaC-Run

## Python

**Run Python in xampp for windows:**

**STEP-1: [Download Python]**

Download & install the latest version of python from www.python.org Download Python & click on the windows installer of any version [ex. python-3.6.2]

**STEP 2: [Install Python]** Install in any directory of your harddrive [ex. D:\python-3.6.2]

**STEP 3: [Configur Python]** Open the directory where xammp was installed Go to apache >> conf [ex. D:\xampp\apache\conf\httpd.conf] You'll see a file named httpd.conf Open it in any text editor & put the below codes in the end of that file

AddHandler cgi-script .py

ScriptInterpreterSource Registry-Strict

**STEP 4:[optional]**

In same file search for When you've found it put index.py in the end It will look something like this

<IfModule dir\_module>

DirectoryIndex index.php index.pl index.cgi index.asp index.shtml index.html index.htm \

default.php default.pl default.cgi default.asp default.shtml default.html default.htm \

home.php home.pl home.cgi home.asp home.shtml home.html home.htm index.py

</IfModule>

**STEP 5:[restart apache/xampp]**

That's all for editing, now restart apache from your xampp control panel

**STEP 6:[Run Python from xammp]**

Open a text editor & test python now on xammp htdoc directory[ex. D:\xampp\htdocs\PythonProject]. But wait at the beginning of your script you need to specify the path where you've installed python. In my case its D:/python-3.6.2/python.exe .In your case it may be different, depending up on the version you've installed python & the directory of your hard drive python Code .

#!D:/python-3.6.2/python.exe

print("Content-Type: text/html\n")

print ("Hello Python Web Browser!! This is cool!!")

Save the file as test.py in htdocs & open <http://localhost/PythonProject>\test.py .If everything goes well, You'll see the text "Hello Python Web Browser!! This is cool!!"

# Networking

|  |  |
| --- | --- |
| **Terms** | **Definition** |
| UTP | Unshielded Twisted Pair Cable |
| Cat-4 | Category 4 ethernet cable |
| RJ-45 | It is connector used in both ends of cat-5 cable |
| Speed, Information | It is usually written on the cable |
| Brown, Blue, Orange, Green | The four pairs of wires in cat-5 cable |
| 1 plain, 1 striped | The combination of wires in pairs in the cat-5 cable |
| Electromagnetic interference, Crosstalk | Cat-5 pairs are twisted to reduce \_\_\_\_\_\_\_ and \_\_\_\_\_\_. |
| Sends data | Orange cat-5 wire pair |
| Receives data | Green cat-5 wire pair |
| Blue, Brown | They are reserved for future bandwidth capacity |
| Solid color | It represents negative polarity |
| Stripe color | It represents positive polarity |
| Bandwidth | The transmission capacity of a computer network or telecommunication system |
| Speed | The rate at which something is able to move. |
| Base T | It is an Ethernet standard |
| Megabits | It is the bandwidth rate used in the telecommunication and computer networking field. |
| Megabyte | It is a data transfer rate used in computing. |
| 568A, 568B | Two RJ-45 connector standards |
| 100 m | It is the maximum length of the cat-5 cable |
| Coaxial cables | It is used to create networks. It has one big copper wire. The different connectors are T connector, end connector, terminator, and coupler. |
| Cat-5 cables | It has 8 wires and RJ-45 are one of the connectors. |
| RJ-62 network | They are also known as Coaxial networks |
| Bus | Coaxial networks count on a central line called \_\_\_\_\_ |
| Toner Tracer | Listens to electrons (noises) generated by the signals |
| Light | Fiber Optic Cables uses \_\_\_\_\_\_ rather electrons to send network information. |
| Cladding | The layer outside of the core of the optical fiber cable. |
| 30 cm | Minimum bend for the optical fiber |
| Microfracture, real fracture, sever leak light | These are the causes of data loss in optical fiber |
| Fusion Splicer | It fixes the broken/overbent fiber optic cables |
| SL, ST, LC | Fiber optic connectors |
| Snap-in connector | SC |
| Straight tip | ST |
| Lucent connector | LC |
| Single mode, Multiple modes | Two types of fiber optic cables |
| Single Path | One path; A laser light used in this fiber optic cables |
| Multiple Mode | A laser light or led light used in this fiber optic cables |
| Water, Heat, Vibration | These are the obstacles of the networking cables |
| J-Hooks | It is used to hang cable from. These are usually attached to floor joint or beams |
| Cable Protectors | These are used to protect cables that have to run on floors. |
| Cable Trays | They are used to hold large numbers of cables that have to run relatively long distances. |
| Raceways | It is often used to run cable to workstations that do not have wall jack access. |
| Cable Ties | It keeps the cables neat. |
| Smurf tubes | It is generally run in new buildings inside the walls. |
| Patch Panel | Organize the able and wire connections to the wiring closet. It works like a telephone switchboard but for cables. |
| Network Topology | This is the picture of \_\_\_\_\_\_\_\_ |
| 568A cable | Which cable is it? |
| 568B cable | Which cable is it? |
| Multimeter | Measures the length of the wire and resistance along a cable |
| Resistance(Ω) | Measures how hard it is for electrons to move through a wire. |
| Higher Resistance | Tougher for electron to move. |
| Infinite Resistance | Broken Wire |
| Long wire | Resistance increases |
| Direct Current | DC (Battery) |
| Alternating Current | AC (Alternating current) |
| Oscilloscope | It shows the voltage changes over time |
| Voltage | Pressure to make electrons move |
| Signal | The \_\_\_\_\_ on a network cable is just the change in voltage |
| Noise | Extraneous Voltage |
| Logic Analyzer | It measure the voltage changes over time. It shows the binary value "1" when the voltage reaches a certain level. It measures whether the signal isn't a strong enough to carry the network data. It interprets the signal into binary data. |
| Lan Analyzer | It has all the functions of the multimeter, logic analyzer and toner tracer. It understands the network traffic in the signal. |
| Frames | The data is in the form of \_\_\_\_\_ frames on an Ethernet network. |
| Non-return zero 1 | NRZ-1 (It is one of the encoding method) Flip the voltage pulses. |
| 0 | Whenever the signal changes from high to low, encode a \_\_\_\_\_ |
| 1 | Whenever the signal changes from low to high, encode a \_\_\_\_\_\_ |
| Network Interface Card (NIC) | Encoding is handled by the \_\_\_\_\_\_\_\_\_\_\_\_ inside the computer. It handles and decodes digital signals, and is in charge of all the messaging ins and outs of the computer. |
| Read-only Memory chips (ROM) | It is in the NIC which stores the MAC address |
| MAC | Media Access Address |
| NRZ coding | Over here the binary data is represented by the high and low voltage levels: high is a 1, low is a 0 |
| Manchester Coding | Over here, it is the transition to a voltage that represents data. The ethernet standard tells hardware how to encode the data. The protocol for 10BaseT Ethernet specifies that the signal will be encoded using \_\_\_\_\_\_\_\_\_. |
| Error Correction | Any time we need to send data on a network, you can run into problems with that data. Different encoding methods allow for detection and correction of those problems. It helps maintain the integrity of the data. |
| 25 | Binary to Decimal (11001) |
| American Standard Code for Information Interchange | ASCII |
| Bit | Each binary digit is called a \_\_\_\_\_ |
| Byte | Eight bits together form a \_\_\_\_\_\_ |
| 42 | Hexadecimal to Decimal (0002A) |
| Protocols | It defines the structure of a message |
| Frame | It is a logical structure of bits that organizes network traffic so every device knows how to read the information inside of it. |
| Packet | It is another structure inside the frame. It tells the correct order (sequence number) |
| Preamble | It is 7 bytes. The regular pattern of bits allows the communicating network devices to synchronize their clock pulses. |
| Start of frame | It is 1 byte. It ends with two '1's. It also indicates that the crucial content is in its way. |
| Destination Mac Address | It is the hardware address of the next network device to which the packet is traveling |
| Source Mac Address | It is the hardware address of the last device that sent the frame |
| Ethertype | It tells what is being transported in the payload |
| Payload | It is the meat of the frame. It holds the data being sent. |
| Cyclic Redundancy Check | It is the number that allows the receiving hardware (the NIC) to check if the frame contains any errors. |
| First Half | \_\_\_\_\_ of the Mac Address is a special code assigned to the manufacturer of the hardware. |
| Last Half | \_\_\_\_\_\_ of the address is a number the manufacturer uses to number the devices they produce |
| UDP (Protocol Type 17) | It is used for streaming data such as music and videos |
| ICMP (Protocol Type 1) | It is used for testing network connections using the ping program |
| TCP Packet (Protocol Type 6) | It is used for most IP network communications that require a reliable connection. (Checking no information is lost) |
| Datagrams | It is used to refer to data sent in packets by an unreliable protocol such UDP or ICMP |
| Mac addresses | It is usually six byte long, or 48 bits. Typically they are written in hexadecimal format and separated by colons or dashes, like this 0f:2b:5d:e7:a3:eb |
| Hubs and switches | \_\_\_\_\_\_\_\_\_\_\_ work on the local area network (LAN) or intranet |
| Routers | It allows us to set up wide area networks (WANs) or internets |
| Hub | It is really just an electrical repeater. It takes whatever signal comes in, and sends it out on all the other ports. It has two devices connected to it that could have sent the signal, a computer and a switch. |
| Switch | It reads the signal as a frame and uses the frame's information to send it where it's supposed to go. This is done by using the MAC address of the frame. They can store packets and forward them to their destinations. |
| Application Specific Integrated Circuits (ASCIS) | These are highly specialized integrated circuits. |
| deletes | If a network device stops transmitting, the switch just \_\_\_\_\_\_\_ the entry to keep the table size small. |
| Wireshark | It is a software that monitors packets |
| IP address | A router looks at the \_\_\_\_\_\_\_\_\_ from the incoming packet and forwards it if it is intended for a workstation located on the other network. It changes the source MAC address to its MAC address and changes the destination MAC address to the workstation the traffic meant for. |
| Physical network | It is the hardware such as the cables, switches, hubs, and routers. |
| Logical network | It is the network addressing stuff. |
| IEEE, manufacturer | 00:A3:03 : 51:OE:AC; first part assigned to the manufacturer by \_\_\_\_\_\_\_\_; second part determined by \_\_\_\_\_\_\_\_\_\_\_\_ |
| A IP address | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is made up of a network address and a host address. 198.168.100 . 1 |
| host | It is the unique bit assigned to a particular network device |
| Subnet mask | 192.168.100.0/24 It tells us that the first 24 bit, or 3 bytes, are the network address and is called the \_\_\_\_\_\_\_\_\_\_\_\_\_. |
| Mac address, address resolution protocol | We retrieve IP addresses using the \_\_\_\_\_\_\_\_and \_\_\_\_\_\_\_\_\_ |
| Ping command | It tells us if thr network and host are reachable or not |
| Traceroute command | It traces the route that the packets take to get to the destination IP address |
| Routing information protocol | RIP. It is a way of sharing network addresses. Routers use RIP to talk to each other, sharing their route information and allowing them to keep their route tables up-to-date |
| Hop-count | It is the number of routers a packet must "hop" through to get to a particular IP network |
| Open Shortest Path First (OSPF) | OSPF |
| Autonomous system number | In order for routers to be neighbors, they must share a common IP subnet and a \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| EIGRP | It uses the diffusing update algorithm |
| Router loop | It is where packets would just go from one router to the next and never get anywhere. They just go around the circle |
| Converge | A network is said to \_\_\_\_\_\_\_\_\_\_ when all the routers have the correct routing information for the network |
| Host name, domain name, fully qualified domain name | www.google.com |
| Domain Name System (DNS) | It translates fully qualified domain names that are meaningful to humans into IP addresses that computers understand |
| Dns zone file | It tells is about name servers, mail server and other servers |
| Reverse DNS | It let's us find a domain for an ip address |
| reverse DNS lookup | Fight against spam in email servers |
| Domain Information Groper (DIG) | Command-line tool in non-Windows systems used to diagnose DNS problems. |
| troubleshooting | the process of analyzing a design problem and finding a way to fix it. Ping, computer to a router, show, interpret the statistics |
| Simple Network Management Protocol (SNMP) | A protocol used to monitor and manage network devices, such as routers, switches, and servers. |
| Management Information Base | The information database of an SNMP managed device |
| Object identifier | The \_\_\_\_\_\_\_\_\_\_ of information in the MIB is called the OID |
| Syslogd daemon | It allows us to have all sorts of network devices send information to that server. That information normally would have being written to a local log file on that device |
| High Priority events | Memory errors, power supply voltages, interface status change |
| Moderate priority events | High network traffic, changes to configuration |
| Low priority events | Unable to reach some websites on internet, low network traffic |
| Radio waves | Wireless networks use \_\_\_\_ with the air being the transmission medium. |
| Dynamic Host Configuration Protocol (DHCP) | An automated mechanism to assign IP addresses and IP configuration information over the network. |
| Network Address Translation (NAT) | Translates the private IP address to a public address for routing over the Internet |
| 802.11n | Boasts a maximum throughput of 600 Mbps. Range is 300m |
| 802.11a | Uses channels in the 5-GHz band and provides a maximum theoretical throughput of 54 Mbps. Range is 35m |
| 802.11b | 11 Mbps. Range is 35m |
| port mapping | Specifying a port in the address table for a particular device, and any traffic that comes in on this port is forwarded to the corresponding device |
| Address Resolution Protocol (ARP) | Part of the TCP/IP protocol for determining the MAC address based on the IP address. |
| Denial of Service (DoS) | An attack that attempts to prevent a system from performing its normal functions. |
| man in the middle | hacker places themselves between client and host |
| MAC flooding | What attack, if successful, causes a switch to function like a hub? |
| Intrusion Detection System (IDS) | A device designed to be active security; it can detect an attack as it occurs. |
| Access Control List (ACL) | The list of permissions granted or denied that is attached to a file or folder. |
| Firewall | Part of a computer system that is designed to block unauthorized access |
| Static Packet Filtering | Firewall that examines header info of every packet. Fast but easy to defeat. |
| stateful packet filter | A packet filter that applies rules to related packets within the same network session. |
| social engineering | using deception to obtain unauthorized access to information resources |
| blueprint | a detailed outline or plan for a building |
| Star Topology | A network topology in which all computers in the network connect to a central wiring point. |
| Bus Topology | The computers share the media (coaxial cable) for data transmission |
| Token Ring Topology | A network topology configured in a logical ring that complements the token passing protocol |
| virtual local area network (VLAN) | Logical subgroup within a LAN created with software instead of hardware. |
| Border Gateway Protocol (BGP) | An ISP protocol that allows routers to share information about routes with each other. |
| Virtual Private Network (VPN) | A connection between two or more computers or devices that are not on the same private network. |

# HTML

## Getting Connected

<http://www.sphotonkhan.com/index.html>

🡨🡪 🡨…………………....🡪🡨…….🡪

The first part of the URL tells us the protocol that needs to be used to retrieve the resource.

The second part is the website name

The third part is the absolute path to the resource from the root path

## Uniform Resource Locator

A **Uniform Resource Locator (URL)** is a global address that can be used to locate anything on the Web, including HTML pages, audio, video, and many other forms of web content. In addition to specifying the location of the resource, a URL also names the protocol that we can use to retrieve the resource

## HTTP

**Hyper Text Transfer Protocol**, it is a simple request and response protocol. It’s an agreed-upon method (a protocol) for transferring hypertext documents around the Web.

## Absolute Path

The path with reference to root directory is called absolute.

<http://www.sphotonkhan.com/index.html>

## Relative Path

The path with reference to current directory is called relative.

<a href = **“../**lounge.html”> To the lounge </a>

“/” means Separate all parts of the path

“..” means parent folder

“../../” means move to parent folders up

## Element

**Element = Openting Tag + Content + Closing Tag**

<h1> Coffee Shop </h1>

Tag consist of the tag name surrounded by angle brackets that is, the < and > characters

## Basic Format

<!DOCTYPE html>

<html>

<head>

<title> Basic Format</title>

<meta charset = "UTF-8"/>

</head>

<body>

<h1> Heading One </h1>

<p> Paragraph </p>

</body>

</html>

**Code 1: Basic.html**

## Comment Tag

**<!-- -->**

The comment tag is used to insert comments in the source code. Comments are not displayed in the browsers.

## Document Type

The **<!DOCTYPE>** declaration must be the very first thing in your HTML document, before the <html> tag.

## Whole Document

<html> element defines the whole document. Standard Markup Language

## Head

**<head>**

<title>Basic Tutorial</title>

<meta charset = “UTF-8”/>

<link rel = “stylesheet” type = “text/css” href = “location.css”/>

<script src = “interaction.js”></script>

**</head>**

## Title

**<title>Basic Tutorial</title>**

The <title> tag is required in all HTML documents and it defines the title of the document.

It defines a title in the browser toolbar

It provides a title for the page when it is added to favorites

It displays a title for the page in search-engine results

## Character Encoding

**<meta charset = “UTF-8”/>**

Meta means we going to tell the browser something about the page

The charset attribute is where we specify the character encoding

UTF-8 is an encoding in the Unicode family of encodings (one of several). “UTF-8” is the version we for the web page

## Link

**<link rel = “stylesheet” type = “text/css” href = “location.css”/>**

Linking the html file to a css file

## Script

**<script src = “interaction.js”></script>**

## Body

The <body> element defines the document body.

## Heading

<h1> Hello </h1> It has in total 6 headings <h1><h2><h3><h4><h5><h6>

## Paragraph

<p> Paragraph </p>

## W3C Validator

http://validator.w3.org/

**Checks the format of the html page**

## Quote

It is a child of paragraph <p> node

**<q> To be or not to be </q>**

## Block Quote

Alternative way to write quotes

**<blockquote> Mr. Mobile have fun today </blockquote>**

|  |  |
| --- | --- |
| **Blockquote <blockquote>** | **Quote <q>** |
| Block quote stands on it own | It is a inline element |

## Block Element vs Inline Elements

|  |  |
| --- | --- |
| **Block Elements** | **Inline Elements** |
| <h1> <h2> <h6> <blockquote> | <q> <a> <em> |
| They can stand on its own | They need to be in block element |

## Class vs Id

|  |  |
| --- | --- |
| **Class** | **Id** |
| Class are not unique | Id are unique |
| You can use the same class on multiple elements. | Each element can have only one id |
| You can use multiple classes on the same element. | You cannot use multiple id in the same class |
| Each page can have multiple element with that class | Each page can have only one element with that id |

## Break

**<br/>**

Breaks the flow and insert a “line break”

## Void Element

Elements that don’t have any content by design are called void elements. When we need to use a void element, like <br> or <img>, we only use an opening tag. This is a convenient shorthand that reduces the amount of markup in our HTML

## Hyperlink

The **<a>** tag defines a hyperlink, which is used to link from one page to another.

Use the <a> element to create a hypertext link to another web page. The content of the <a> element becomes clickable in the web page. The

The href attribute tells the browser the destination of the link

The title attribute gives the textual description of the page

**<a href = “elixir.html” title = “Description of the page”>Elixirs</a>**

href is the attribute

### Create Destination

The id attribute allows us to uniquely identify an element

# on the end of our lonk follow the destination identifier

<h2 **id = “books”**> Books </h2>

<a href = “**index.html#books**”> See Books </a>

### New window using target

<a **target = “\_blank”** href = <http://sphotonkhan.com>

title = Read all about caffeine on the site> Photon Khan</a>

The target attribute tells the browser where to open the page that is at the link the href attribute. If there is not target, then the browser opens the link in the same window. If the target is “\_blank”, then the browser opens the link in a new window.

## Image

**<img src = “drinks.gif” alt = “Drinks Page”/>**

The src attribute is the location of the image

The alt attribute is the alternative description of the image

|  |  |  |
| --- | --- | --- |
| JPEG | PNG | GIF |
| Works best for continuous tone images, like photographs | It works best for images with a few solid colors, and image with lines, like logos, clip art, and small text in images | GIF works best for few solid colors and images with lines, logos, clip art and small text in images |
| It represents 16 million different colors | It represents images with millions of different colors. PNG comes in three different formats  PNG-8, PNG-24, PNG-32  Depending on color representation | It represents 256 different colors |
| It is a “lossy” format because to reduce the file size, it throws away some information about the image | PNG compresses the file to reduce its size, it is not a lossy format | It is a lossless format |
| It does not support transparency | It supports transparency | It supports transparency |
| Files are smaller for more efficient web pages | Files tend to be larger than JPEG but can be smaller or larger than GIF depending on the colors used | GIF size larger than JPEG |
| No support for animation |  | Support animation |

### Sizing Image

<img src = “tea.gif” alt = “Drinks Page” **width = “30” height = “23”**/>

### Pixel

A good rule of thumb is **96 pixels** to every inch. Browser fits **1200 x 800 pixels (Average)**

## Link out of images

<**a** href = "http://sphotonkhan.com" target = "\_blank" title = "Cafe">

<**img** src = "images/computer.ico" alt = "computer icon"/>

</**a**>

If we are placing transparent image in our web page, make sure the matte color of the image matches the background color of our web page. We can use PNG or GIF format for our transparent image.

## Ordered List

The **<ol>** tag defines an ordered list. An ordered list can be **numerical or alphabetical.**

<ol>

<li> Bread </li>

<li> Egg </li>

<li> Jam </li>

</ol>

## Unordered List

The **<ul>** tag defines an unordered **(bulleted)** list.

Use the <ul> tag together with the <li> tag to create unordered lists.

<ul>

<li> Bread </li>

<li> Egg </li>

<li> Jam </li>

</ul>

## Nesting

Putting one element inside another

<**ol**>

<**li**> Charge Segway </**li**>

<**ul** Pack for trip>

<**li**>Cell Number</**li**>

<**li**>iPod</**li**>

<**li**>Digital Camera</**li**>

<**li**>Protein Bar</**li**>

</**ul**>

</**ol**>

## Definition List

<**dl**>

<**dt**>Byrna Shave Signs</**dt**>

<**dd**> Road Signs common</**dd**>

<**dt**>Route 6666</**dt**>

<**dd**>Most Famous Guys</**dd**>

</**dl**>

The <dd> tag is used to describe a term/name in a description list.

The <dd> tag is used in conjunction with <dl> (defines a description list) and <dt> (defines terms/names).

Inside a <dd> tag you can put paragraphs, line breaks, images, links, lists, etc.

## Emphasize

The <em> tag is a phrase tag. It renders as emphasized text.

**<p><em>** Paragraph **</em></p>**

## Strong

This element is used to mark up text we want emphasized with extra strength

<**li**><**strong**> Charge Segway </**strong**></**li**>

## Preformatted text

The **<pre>** tag defines preformatted text.

Text in a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.

<pre>

Text in a pre element

is displayed in a fixed-width

font, and it preserves

both spaces and

line breaks

</pre>

## Formatting elements

**<i>** - Italic text

**<mark>** - Marked text

**<small>** - Small text

**<del>** - Deleted text

**<ins>** - Inserted text

**<sub>** - Subscript text

**<sup>** - Superscript text

## Time

The element tells the browser that the context is a date or time, or both

<p> We open at **<time>10:00</time>** every morning. </p>

## Code

The <code> tag is a phrase tag. It defines a piece of computer code.

**<code>A piece of computer code</code>**

## Center

**<center>**

<font size = “small”>You got to be kidding me</font>

**</center>**

## Font

**<font size = “small”>You got to be kidding me</font>**

**<font face = “arial”>You got to be kidding me</font>**

## Special Attributes

### Text Align Attribute

<p **align=right**>Come over here</p>

No double apostrophe for the attribute

### Color Attribute

<body **bgcolor = “tan” text = “black”**>

## Division

This helps to organize the elements belong to the same group. This structure helps us separate a page into a logical sections for clarity and styling.

The <div> tag defines a division or a section in an HTML document.

The <div> element is often used as a container for other HTML elements to style them with CSS or to perform certain tasks with JavaScript.

<div style="background-color:lightblue">  
  <h3>This is a heading</h3>  
  <p>This is a paragraph.</p>  
</div>

## Span

The <span> tag is used to group inline-elements in a document.

The <span> tag provides no visual change by itself.

The <span> tag provides a way to add a hook to a part of a text or a part of a document.

<p>My mother has <span style="color:blue">blue</span> eyes.</p>

## Header

The <header> element represents a container for introductory content or a set of navigational links.

A <header> element typically contains:

* One or more heading elements (<h1> - <h6>)
* Logo or icon authorship information
* We can have several <header> elements in one document.

<header>  
    <h1>Most important heading here</h1>  
    <h3>Less important heading here</h3>  
    <p>Some additional information here</p>  
</header>

## Footer

The <footer> tag defines a footer for a document or section.

A <footer> element should contain information about its containing element.

A <footer> element typically contains:

* Authorship information
* Copyright information
* Contact information
* Sitemap
* Back to top links
* Related documents

You can have several <footer> elements in one document.

<footer>  
  <p>Posted by: Hege Refsnes</p>  
  <p>Contact information: <a href="mailto:someone@example.com">  
  someone@example.com</a>.</p>  
</footer>

## Aside

The <aside> tag defines some content aside from the content it is placed in.

The aside content should be related to the surrounding content.

<aside>  
  <h4>Epcot Center</h4>  
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>  
</aside>

## Section

It defines sections in a document, such as chapters, headers, footers, or any other sections of the document.

<section>  
  <h1>WWF</h1>  
  <p>The World Wide Fund for Nature (WWF) is....</p>  
</section>

## Navigation

It defines group of navigation links

<nav>  
  <a href="/html/">HTML</a> |  
  <a href="/css/">CSS</a> |  
  <a href="/js/">JavaScript</a> |  
  <a href="/jquery/">jQuery</a>  
</nav>

## Video

<video controls autoplay width=”512” height=”288”

src=”video/tweetsip.mp4”

poster=”images/poster.png”>

</video>

**autoplay** Specifies that the video will start playing as soon as it is ready

**controls** Specifies that video controls should be displayed (such as a play/pause button etc).

**height** Sets the height of the video player

**loop** Specifies that the video will start over again, every time it is finished

**muted** Specifies that the audio output of the video should be muted

**poster** Specifies an image to be shown while the video is downloading, or until the user hits the play button

**preload** Specifies if and how the author thinks the video should be loaded when the page loads

**src** Specifies the URL of the video file

**width** Sets the width of the video player

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

The HTML5 specification allows for any video format. It is the browser implementation that determines what formats are actually supported

### Video Format Contenders

|  |  |  |
| --- | --- | --- |
| **MP4 container with H.264 and AAC audio** | **WebM container with VP8 video and Vorbis audio** | **Ogg container with Theora video and Vorbis audio** |
| H.264 is licensed by the MPEG-LA group | WebM was designed by Google to work with VP8-encoded videos. | Theora is a open source codec |
| There is more than one kind of H.264, each is known as profile | WebM/VP8 is supported by Firefox, Chrome, and Opera | Video encoded with Theora is usually contained in an Ogg file, with the .ogv file extension |
| MP4/H.274 is supported by Safari and IE9+. We can find support in some version of chrome | WebM-formatted videos with the.webm extension | Ogg/Theora is supported by Firefox, Chrome and Opera |

**Video with specific format**

<video src = “video/tweetsip.ogv” type = “video/ogg; codecs=’theora, vorbis’”>

## Audio

It defines sound, such as music or other audio streams.

<audio controls>  
  <source src="horse.ogg" type="audio/ogg">  
  <source src="horse.mp3" type="audio/mpeg">  
  Your browser does not support the audio tag.  
</audio>

**autoplay** Specifies that the audio will start playing as soon as it is ready

**control**  Specifies that audio controls should be displayed (such as a play/pause button etc)

**loop**  Specifies that the audio will start over again, every time it is finished

**muted**  Specifies that the audio output should be muted

**preload** Specifies if and how the author thinks the audio should be loaded when the page loads

**src** Specifies the URL of the audio file

## Object

The defines an embedded object within an HTML document. Use this element to embed multimedia (like audio, video, Java applets, ActiveX, PDF, and Flash) in your web pages.

<object width="400" height="400" data="helloworld.swf"></object>

It is also embedded to another webpage into your HTML document.

We can use the <param> tag to pass parameters to plugins that have been embedded it

## Meter

It defines a scalar measurement within a known range, or a fractional value. This is also known as a gauge.

<meter value="2" min="0" max="10">2 out of 10</meter><br>  
<meter value="0.6">60%</meter>

**form** Specifies one or more forms the <meter> element belongs to

**high** Specifies the range that is considered to be a high value

**low** Specifies the range that is considered to be a low value

**max** Specifies the maximum value of the range

**min** Specifies the minimum value of the range

**optimum** Specifies what value is the optimal value for the gauge

**value** Required. Specifies the current value of the gauge

## Canvas

It is used to draw graphics, on the fly, via scripting (usually JavaScript).

It is only a container for graphics, you must use a script to actually draw the graphics.

<canvas id="myCanvas"></canvas>

## Figure

It specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

<figure>  
  <img src="img\_pulpit.jpg" alt="The Pulpit Rock" width="304" height="228">  
</figure>

## Progress

It represents the progress of a task.

<progress value="22" max="100"></progress>

## Table

It defines an HTML Table

<table>  
  <tr>  
    <th>Month</th>  
    <th>Savings</th>  
  </tr>  
  <tr>  
    <td>January</td>  
    <td>$100</td>  
  </tr>  
</table>

<**html**>

<**head**>

<**title**> Basic Format</**title**>

<**meta** charset = "UTF-8"/>

<**style** type = "text/css">

**table**, **td**, **th**{

**border**: **thin** **solid** black;

**border-collapse**: **collapse**;

}

</**style**>

</**head**>

<**body**>

<**table**>

<**tr**>

<**th**> City </**th**>

<**th**> Date </**th**>

<**th**> Temperature </**th**>

<**th**> Altitude </**th**>

<**th**> Population </**th**>

<**th**> Diner Rating </**th**>

</**tr**>

<**tr**>

<**td**> Walla Walla</**td**>

<**td**> June 4th</**td**>

<**td**> 75</**td**>

<**td**> 1,204 </**td**>

<**td**> 29.686 </**td**>

<**td**> 4/5 </**td**>

</**tr**>

<**tr**>

<**td** rowspan = "2"> Truth or Consequence </**td**>

<**td**> August 9th </**td**>

<**td** colspan = "2"> 93 </**td**>

<**td** rowspan = "2"> 7.289 </**td**>

<**td**> 5/5 </**td**>

</**tr**>

<**tr**>

<**td**> August 27th </**td**>

<**td**> 98 </**td**>

<**td**> 4,242 </**td**>

<**td**> 4/5 </**td**>

</**tr**>

<**tr**>

<**td**> Why </**td**>

<**td**> August 18th </**td**>

<**td**> 104 </**td**>

<**td**> 860 </**td**>

<**td**> 480 </**td**>

<**td**> 3/5 </**td**>

</**tr**>

</**table**>

</**body**>

</**html**>

## Form

<**form** enctype = "multipart/form-data" method = "post" action = "">

<**label** for = "fullname"> Full Name </**label**><**br**/>

<**input** type = "text" name = "fullname" placeholder = "Your Name" required/><**br**/><**br**/>

<**label** for = "email"> Email </**label**><**br**/>

<**input** type = "email" name = "email" placeholder = "Your Email"required/><**br**/><**br**/>

<**label** for = "password"> Password </**label**><**br**/>

<**input** type = "password" name = "password" required><**br**/><**br**/>

<**label** for = "age"> Age </**label**><**br**/>

<**input** type = "number" name = "age" required/><**br**/><**br**/>

<**label** for = "gender"> Gender </**label**><**br**/>

Male <**input** type = "radio" name = "gender" value = "male">

Female <**input** type = "radio" name = "gender" value = "female"><**br**/><**br**/>

<**fieldset** height = "100px" width = "100px">

<**legend** for = "spice"> Spice </**legend**><**br**/>

Salt <**input** type = "checkbox" name = "spice[]" value = "salt"><**br**/>

Pepper <**input** type = "checkbox" name = "spice[]" value = "pepper"><**br**/>

Garlic <**input** type = "checkbox" name = "spice[]" value = "garlic"><**br**/>

</**fieldset**><**br**/><**br**/>

<**label** for = "car"> Cars </**label**><**br**/>

<**select** name = "car" multiple>

<**option** value = "toyota">Toyota</**option**>

<**option** value = "mazda">Mazda</**option**>

<**option** value = "nissan">Nissan</**option**>

</**select**><**br**/><**br**/>

<**label** for = "distance">Distance Travelled</**label**><**br**/>

<**input** type = "range" min = "5" max = "30" step = "5"><**br**/><**br**/>

<**label** for = "date">Date Today </**label**><**br**/>

<**input** type = "date" name = "date"/><**br**/><**br**/>

<**label** for = "cellphone"> Cellphone </**label**><**br**/>

<**input** type = "tel" name = "cellphone"><**br**/><**br**/>

<**label** for = "fav\_color"> Favorite Color </**label**><**br**/>

<**input** type = "color" name = "fav\_color"/><**br**/><**br**/>

<**label** for = "website"> Website Name </**label**><**br**/>

<**input** type = "url" name = "website"><**br**/><**br**/>

<**label** for = "feedback"> Feedback </**label**><**br**/>

<**textarea** name = "feedback" rows = "10" cols = "48"></**textarea**><**br**/><**br**/>

<**input** type = "submit" name = "submit" value = "Confirm"/>

</**form**>

**<form>** Write a form

**<input>** Take different types of input

Types 🡪 email, password, range, color, tel, submit, checkbox, radio

**<textarea>**

## HTML TERMS

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Description** | **Element** | **Description** |
| <!-- --> | Comment | <br/> | Break |
| <!DOCTYPE> | Doc Declaration | <ul> | Unordered List |
| <a> | Hyperlink | <ol> | Ordered List |
| <img> | Image | <li> | List |
| <q> | Quote | <em> | Emphasize |
| <blockquote> | Blockquote | <strong> | Strong Emphasize |
| <h1> 🡪 <h6> | Heading 1 🡪 6 | <dd> | Describe Data |
| <p> | Paragraph | <dl> | Description List |
| <pre> | Preformatted text | <dt> | Define Terms |
| <time> | Date or Time | <code> | Computer Code |
| <i> | Italic text | <mark> | Marked text |
| <small> | Small text | <del> | Deleted text |
| <ins> | Inserted text | <sub> | Subscript text |
| <sup> | Superscript text | <center> | Center script |
| <meta> | Character encoding | <font> | Font |
| <link> | Link Css File | <script> | Include JavaScript |
| <div> | Division | <title> | Title |
| <span> | Add a hook/Error | <header> | Header |
| <footer> | Footer | <aside> | Sidebar |
| <section> | Section | <nav> | Navigation Bar |
| <video> | Video Element | <audio> | Audio Element |
| <canvas> | Canvas Container | <figure> | Contains diagrams |
| <table> | Table | <progress> | Shows progress bar |
| <th> | Table Heading | <tr> | Table Row |
| <td> | Table Data | <input | User Input |
| <form> | User Form | <fieldset> | Creates a Field |
| <legend> | Subtopic Region | <caption> | Caption |

# CSS

## Basic Format

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



The selector points to the HTML element we want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

## Jigsaw Validator

http://jigsaw.w3.org/css-validator/

Checks the format of css

## Overriding Inheritance

body {font-family: sans-serif;}

h1, h2 {color: gray;}

p {color: maroon;}

## Selector

Select more than one element

h1, h2{color: gray;}

### Class Selector

<p class = “greentea”>Green Tea<p>

p.greentea{color: green;}

#### Multiple Class Selector

We might have a **conflict** while using multiple classes.

<p class = “greentea raspberry blueberry”>Green Tea<p>

<p class = “raspberry”> Raspberry Pie</p>

p.greentea{color: green;}

### ID Selector

<p id = “greentea”>Green Tea<p>

p#greentea{color: green;}

## Attribute Selector

img[width] {border: black thin solid;}

Selects all images that have a width attribute in their HTML

img[height=”300”] {border: red thin solid;}

This selects all images that have a height attribute with a value of 300

image[alt~=”flower”]{border: #ccc thin solid}

This selects all images that have an alt attribute that includes the word “flowers”

## Selecting by Sibling

h1 + p {font-style: italic}

Selects all the paragraphs that comes immediately after an <h1> element

## Combining Selectors

.blueberry p {color: purple}

Select all paragraphs that are descendants of an element in the blue berry class

div#greentea > blockquote

Descendant selector with a <div> with am id “greentea” must be the parentt of the <blockquote>

div#greentea > blockquote p

🡨………………Context……………………🡪🡨Element🡪

<p> element must be a descendant of <blockquote> which must be a child of a <div> with an id of “greentea”

div#greentea > blockquote p: first-line{font-style: italic}

🡨………………Context……………………🡪 🡨E🡪🡨Pseudo Element🡪

<p> element must be a descendant of <blockquote> which must be a child of a <div> with an id of “greentea” and first line of the paragraph must be italic

## Color

The color property specifies the color of text.

h1 {color: #00ff00;}

List of fixed colors: aqua, black, blue, gray, green, lime, maroon, navy, olive, purple, red. silver, teal, white, yellow

### Hex Code Color Calculation

# CC 66 00

🡨Red🡪 🡨Green🡪 🡨Blue🡪

For Red,

12\*16 + 16 = 208

For Green,

6\*16 + 6 = 102

For Blue,

0\*16 + 0 = 0

|  |  |  |
| --- | --- | --- |
| **Decimal (base 10)** | **Binary** | **Hexa (Base 16)** |
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0111 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| **10** | **1010** | **A** |
| **11** | **1011** | **B** |
| **12** | **1100** | **C** |
| **13** | **1101** | **D** |
| **14** | **1110** | **E** |
| **15** | **1111** | **F** |

## Background

The background shorthand property sets all the background properties in one declaration.

body {background: lightblue url("img\_tree.gif") no-repeat fixed center}

### background-color

The background-color property sets the background color of an element.

body {background-color: #92a8d1;}

body {background-color: rgb(201, 76, 76);}

body {background-color: rgba(201, 76, 76, 0.3);}

body {background-color: hsl(89, 43%, 51%);}

body {background-color: hsla(89, 43%, 51%, 0.3);}

### background-image

The background-image property sets one or more background images for an element.

By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.

body {background-image: url("img\_tree.gif"), url("paper.gif");}

### background-position

The background-position property sets the starting position of a background image.

body {background-position: center;}

left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom

x%, y%

xpos, ypos

### background-size

The background-size property specifies the size of the background images.

#example1 {background-size: auto;}

#example2 {background-size: 300px 100px;}

("auto", "cover" and "contain"), the one-value syntax (sets the width of the image (height becomes "auto"), the two-value syntax (first value: width of the image, second value: height), and the multiple background syntax (separated with comma).

**auto** Default value. The background image is displayed in its original size

**cover** Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges

**contain** Resize the background image to make sure the image is fully visible

### background-repeat

The background-repeat property sets if/how a background image will be repeated.

The background image is repeated both vertically and horizontally. The last image will be clipped if it does not fit. This is default

body {background-repeat: repeat-y}

body {background-repeat: no-repeat}

**repeat-x** The background image is repeated only horizontally

**repeat-y** The background image is repeated only vertically

**no-repeat** The background-image is not repeated.

**space** The background-image is repeated as much as possible without clipping.

**round** The background-image is repeated and squished or stretched to fill the space

### background-origin

The background-origin property specifies the origin position (the background positioning area) of a background image.

#example1 {background-origin: content-box;}

**padding-box** The background image starts from the upper left corner of the padding edge

**border-box** The background image starts from the upper left corner of the border

**content-box** The background image starts from the upper left corner of the content

### background-clip

The background-clip property defines how far the background (color or image) should extend within an element.

div {background-clip: padding-box;}

### background-attachment

The background-attachment property sets whether a background image scrolls with the rest of the page, or is fixed.

body {background-attachment: fixed;}

## Box Model

margin

Border

Padding

Content

## Border

Defines a border around the body that is dotted and the color black

h2 {border: 4px dotted blue;}

**border-width** Specifies the width of the border. Default value is "medium"

**border-style** Specifies the style of the border. Default value is "none"

**border-color** Specifies the color of the border. Default value is the color of the text

table{border-collapse: collapse} Collapses two lines into one

### Border Style

The border-style property sets the style of an element's four borders. This property can have from one to four values.

div {border-style: dotted;}

**border-style: dotted solid double dashed**

top border is dotted

right border is solid

bottom border is double

left border is dashed

**border-style: dotted solid double;**

top border is dotted

right and left borders are solid

bottom border is double

**border-style: dotted solid;**

top and bottom borders are dotted

right and left borders are solid

**border-style: dotted;**

all four borders are dotted

**hidden**  The same as "none", except in border conflict resolution for table elements

**dotted** Specifies a dotted border

**dashed** Specifies a dashed border

**solid** Specifies a solid border

**double** Specifies a double border

**groove** Specifies a 3D grooved border. The effect depends on the border-color value

**ridge** Specifies a 3D ridged border. The effect depends on the border-color value

**inset** Specifies a 3D inset border. The effect depends on the border-color value

**outset** Specifies a 3D outset border. The effect depends on the border-color value

### Border Color

The border-color property sets the color of an element's four borders.

div {border-color: coral;}

### Border Width

The border-width property sets the width of an element's four borders.

div {border-width: thin;}

**medium** Specifies a medium border. This is default

**thin** Specifies a thin border

**thick** Specifies a thick border

### Specifying Border Sides

div {border-top-color: coral;}

div {border-top-style: dotted;}

div {border-top-width: thin;}

div {border-right-color: coral;}

div {border-right-style: dotted;}

div {border-right-width: thin;}

div {border-bottom-color: coral;}

div {border-bottom-style: dotted;}

div {border-bottom-width: thin;}

div {border-left-color: coral;}

div {border-left-style: dotted;}

div {border-left-width: thin;}

## Margin

The margin property sets the margins for an element.

p {margin: 35px;}

margin-top, margin-right, margin-bottom, margin-left

## Padding

An element's padding is the space between its content and its border.

p {padding: 35px 70px 50px 90px;}

35 pixels for top, 70 pixels for right, 50 pixels for bottom, and to 90 pixels for left:

p {padding: 35px 70px 50px}

35 pixels for top, 70 pixels for right and left, and to 50 pixels for bottom:

p {padding: 35px 70px}

35 pixels for top and bottom, and to 70 pixels for right and left:

p {padding: 35px}

Padding for all four sides of a <p> element to 35 pixels:

## Top

The top property affects the vertical position of a positioned element. This property has no effect on non-positioned elements.  
div {  
    position: absolute;  
    top: 50px;  
    border: 3px solid green;  
}

**auto** Lets the browser calculate the top edge position. This is default

**length** Sets the top edge position in px, cm, etc. Negative values are allowed.

**%** Sets the top edge position in % of containing element. Negative values are allowed

## Font-Family

The font-family property specifies the font for an element.

The font-family property can hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.

There are two types of font family names: **family-name, generic-family**

p.a {font-family: "Times New Roman", Times, serif;}

There are five font family-name: **sans-serif, serif, monospace, cursive, and fantasy**

Sans-serif family: Verdana, Arial Black, Trbuchet MS, Arial, Geneva

Serif-family: Times, Times New Roman, Georgia

Monospace family: Courier, Courier New, Andale Mono

Cursive Family: Comic Sans, Apple Chancery

Fantasy family: Last Ninja, Impact

p.a {font-family: Verdana, Geneva, Arial, sans-serif;}

If Verdana is not available use Geneva, If Geneva not available use Arial or else sans-serif

**Shortcut**

font: font-style font-variant font-weight font-size/line0height font-family

.a {font: small/1.6em Verdana, Helvetica, Arial, sans-serif;}

## Font Face

With the @font-face rule, web designers do not have to use one of the "web-safe" fonts.

In the @font-face rule you must first define a name for the font (e.g. myFirstFont), and then point to the font file.

@font-face {  
    font-family: myFirstFont;  
    src: url(sansation\_bold.woff);  
    font-weight: bold;  
}

**Formats of Fonts**

TrueType fonts: **.ttf**

OpenType fonts: **otf**

Embedded OpenType fonts: **.eot**

SVG fonts: **.svg**

Web open font format: **.woff**

## Font Weight

The font-weight property sets how thick or thin characters in text should be displayed.

p.thick {font-weight: bold;}

**normal** Defines normal characters

**bold**  Defines thick characters

**bolder** Defines thicker characters

**lighter**  Defines lighter characters

**100, 200, 300, 400, 500, 600, 700, 800, 900**

## Font Style

The font-style property specifies the font style for a text.

p.a {font-style: normal;}

**normal** The browser displays a normal font style. This is default

**italic**  The browser displays an italic font style

**oblique** The browser displays an oblique font style

## Font Size

The font-size property sets the size of a font.

div.a {font-size: 15px;}  
div.b {font-size: large;}  
div.c {font-size: 150%;}

## Text Decoration

The text-decoration property specifies the decoration added to text, and is a shorthand property for: text-decoration-line (required), text-decoration-color, text-decoration-style

h1 {text-decoration: overline;}  
h2 {text-decoration: line-through}  
h3 {text-decoration: underline;}  
h3 {text-decoration: underline overline}

h1 {text-decoration: underline overline dotted red;}  
h2 {text-decoration: underline overline wavy blue;}

## Line Height

The line-height property specifies the height of a line.

div.a {line-height: normal;}  
div.b {line-height: 1.6;}  
div.c {line-height: 80%;}  
div.d {line-height: 200%;}

div.b {line-height: 1.6em;}

em 🡪 1.6 times more than the inherited font size

## List Style

The list-style property is a shorthand for the following properties:

ul {list-style: square inside url("sqpurple.gif");}

list-style-type

list-style-position

list-style-image

### List Style Image

It replaces the list-item marker with an image

ul {list-style-image: url('sqpurple.gif');}

## Text Alignment

The text-align property specifies the horizontal alignment of text in an element.

div.a {text-align: center;}  
div.b {text-align: left}  
div.c {text-align: right;}  
div.c {text-align: justify;}

**justify** Stretches the lines so that each line has equal width (like in newspapers and magazines)

## Letter Spacing

The letter-spacing property increases or decreases the space between characters in a text.

h1 {letter-spacing: 3px;}

**normal** No extra space between characters. This is default

**length** Defines an extra space between characters (negative values are allowed).

## Media Queries

Media queries area an are of active development by the standards groups.

The @media rule is used in media queries to apply different styles for different media types/devices.

@media only screen and (max-width: 600px) {  
    body {  
        background-color: lightblue;  
    }  
}

@media only screen and (orientation: landscape) {  
    body {  
        background-color: lightblue;  
    }  
}

/\* When the width is between 600px and 900px OR above 1100px - change the appearance of <div> \*/  
@media screen and (max-width: 900px) and (min-width: 600px), (min-width: 1100px) {  
  div.example {  
    font-size: 50px;  
    padding: 50px;  
    border: 8px solid black;  
    background: yellow;  
  }  
}

Media queries can be used to check many things, such as:

* Width and height of the viewport
* Width and height of the device
* Orientation (is the tablet/phone in landscape or portrait mode
* Resolution

<link rel="stylesheet" media="screen and (min-width: 900px)" href="widescreen.css">

<link rel="stylesheet" media="screen and (max-width: 600px)" href="smallscreen.css">

## Pseudo Class

a:link{color: green}

The selector is applied to links when they are in an unvisited state

a:visited{color: red}

This selector is applied to links when they are visited

a:hover{color: yellow}

This selector is applied when we hover over a link

p:nth-child(even){background-color:red;}

p:nth-child(2n){background-color:red;}

p:nth-child(odd){background-color:green;}

p:nth-child(2n + 1){background-color:green;}

## Pseudo Elements

p:first-letter{font-size: 3em}

First letter of the paragraph is becoming large

p:first-line{font-style: italic}

First line is becoming italic

## Cascading Styling Sheets

0 0 0

**First Digit:** Does this selector have any ids. One point for each

**Second Digit:** Does this selector have any classes or pseudo-classes. One point for each

**Third Digit:** Does this selector have any element names. One point for each

Override a style using

h1{font-size: 200% **!important**;}

**Specificity Point Examples**

|  |  |  |
| --- | --- | --- |
| h1.greentea | 011 |  |
| p img | 002 |  |
| a:link | 011 |  |
| ol li p | 003 |  |
| .green | 011 |  |
| #elixirs h1 | 102 | Winner |
| em | 001 |  |
| span.cd | 011 |  |
| #sidebar | 101 |  |

Reorder the rules, from highest score to lowest score in CSS

## Float

The float property specifies how an element should float.

Note: Absolutely positioned elements ignores the float property!

img {  
    float: right;  
}

## Position

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

h2 {  
    position: absolute;  
    left: 100px;  
    top: 150px;  
}

**static** Default value. Elements render in order, as they appear in the document flow

**absolute** The element is positioned relative to its first positioned (not static) ancestor element

**fixed** The element is positioned relative to the browser window

**relative** The element is positioned relative to its normal position,

**sticky** The element is positioned based on the user's scroll position

#parent1 {  
    position: static;  
    border: 1px solid blue;  
    width: 300px;  
    height: 100px;  
}  
  
#child1 {  
    position: absolute;  
    border: 1px solid red;  
    top: 70px;  
    right: 15px;  
}  
  
#parent2 {  
    position: relative;  
    border: 1px solid blue;  
    width: 300px;  
    height: 100px;  
}

#child2 {  
    position: absolute;  
    border: 1px solid red;  
    top: 70px;  
    right: 15px;  
}

## Display

The display property specifies the display behavior (the type of rendering box) of an element.

p.ex1 {display: none;}  
p.ex2 {display: inline;}  
p.ex3 {display: block;}  
p.ex4 {display: inline-block;}

**inline** Displays an element as an inline element

Any height and width properties will have no effect

**block** Displays an element as a block element

It starts on a new line, and takes up the whole width

**flex** Displays an element as a block-level flex container

**grid** Displays an element as a block-level grid container

**inline**-**block** Displays an element as an inline-level block container.

The element itself is formatted as an inline element.

We can apply height and width values

**inline**-**flex** Displays an element as an inline-level flex container

**inline**-**grid** Displays an element as an inline-level grid container

**inline**-**table** The element is displayed as an inline-level table

**list**-**item** Let the element behave like a <li> element

**run**-**in** Displays an element as either block or inline, depending on context

**table** Let the element behave like a <table> element

**table**-**caption** Let the element behave like a <caption> element

**table**-**column**-**group** Let the element behave like a <colgroup> element

**table**-**header**-**group** Let the element behave like a <thead> element

**table**-**footer**-**group** Let the element behave like a <tfoot> element

**table**-**row**-**group** Let the element behave like a <tbody> element

**table**-**cell** Let the element behave like a <td> element

**table**-**column** Let the element behave like a <col> element

**table**-**row** Let the element behave like a <tr> element

## Vertical Align

The vertical-align property sets the vertical alignment of an element.

img.a {vertical-align: baseline;}  
img.b {vertical-align: text-top;}  
img.c {vertical-align: text-bottom;}  
img.d {vertical-align: sub;}  
img.e {vertical-align: super;}

## Vendor Specific CSS Properties

-moz-transform

Moz stands for mozilla

div{

transform: rotate(45deg), **(General)**

-webkit-transform: rotate(45deg), **Safari & Chrome**

-moz-transform: rotate(45deg), (**Mozilla)**

-ms-transform: rotate(45deg), **(Internet Explorer)**

-o-transform: rotate(45deg) **(Opera)**

}

## CSS TERMS

|  |  |  |  |
| --- | --- | --- | --- |
| background | text-align | @media | -webkit-transform |
| border | letter-spacing | border-style | -ms-transform |
| color | font-weight | border-color | -o-transform |
| margin | font-style | table |  |
| padding | list-style | table-cell |  |
| font-family | line-height | table-row |  |
| position | font-size | vertical-align |  |
| top | @font-face | -moz-transform |  |

# JavaScript

## Basics

### Basic Structure

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Basic File </**title**>

<**meta** charset = "UTF-8">

<**script** src = "wakeup.js"></**script**>

</**head**>

<**body**></**body**>

</**html**>

setTimeout(wakeUpUser, 2000);

**function** wakeUpUser(){

alert("Get Started With Javascript");

}

**Code 1: wakeupuser.js**

### Document Write

This method writes HTML expressions or JavaScript code to a document.

**var** count = 4;

**for**(**var** i = 0; i < count; i++){

document.write(i + "<br/>");

}

**Code 2: documentwrite.js**

This method is mostly used for testing: If it is used after an HTML document is fully loaded, it will delete all existing HTML. This method should be heavily avoided

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p>My first paragraph.</p>

<button type="button" onclick="myFunction()">Click me!</button>

<script>

function myFunction() {

document.write("Hello World");

}

</script>

</body>

</html>

**Code 3: documentwriteavoid.html**

Open an output stream, add some text, then close the output stream:

<!DOCTYPE html>

<**html**>

<**body**>

<**p**>Click It.</**p**>

<**button** onclick="myFunction()">Try it</**button**>

<**script**>

**function** myFunction() {

document.open();

document.write("<h1>Hello World</h1>");

document.close();

}

</**script**>

</**body**>

</**html**>

**Code 4: documentwriteoutput.html**

Open a new window called "MsgWindow", and write some text into it:

<!DOCTYPE html>

<**html**>

<**body**>

<**p**>Click Here</**p**>

<**button** onclick="myFunction()">Try it</**button**>

<**script**>

**function** myFunction() {

**var** myWindow = window.open("", "MsgWindow", "width=200,height=100");

myWindow.document.write("<p>This is 'MsgWindow'. I am 200px wide and 100px tall!</p>");

}

</**script**>

</**body**>

</**html**>

**Code 5: documentwritewindow.html**

### Console Log

The console.log() method writes a message to the console.

The console is useful for testing purposes.

### Debugger

The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function.

This has the same function as setting a breakpoint in the debugger.

If no debugging is available, the debugger statement has no effect.

**var** a = 3;

**var** b = 5;

console.log(a);

console.log(b);

**debugger**;

console.log(a + b);

**Code 6: console.js**

### Time Out

This setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds. Therefore, 1000 ms = 1 second. This function is only executed once.

Use the clearTimeout() method to prevent the function from running.

setTimeout(**function**(){

alert("Wake up!")

}, 3000);

**Code 7: timeout.js**

### Prompt

The prompt() method displays a dialog box that prompts the visitor for input.

**var** getInput = prompt("Give me your input bro!");

console.log(getInput);

**Code 8: prompt.js**

## Math

*/\*\**

*\* Give any decimal value between 0 to 1*

*\*/*

**var** random = Math.random();

console.log(random);

*/\*\**

*\* Give any decimal value between 0 to 5*

*\*/*

**var** random\_five = (5 \* Math.random());

console.log(random\_five);

*/\*\**

*\* Math Ceil*

*\*/*

console.log(Math.ceil(random\_five));

*/\*\**

*\* Math Floor*

*\*/*

console.log(Math.floor(random\_five));

*/\*\**

*\* Math Square Root*

*\*/*

**var** number = 9;

console.log(Math.sqrt(number));

*/\*\**

*\* Math PI*

*\*/*

radius = 3;

console.log("Area of the Circle: " + (Math.PI \* Math.pow(3,2)))

**Code 9: math.js**

## Function

Key Point: If we forget to declare a variable before using it, the variable will always be global (even if first time we use it is in a function.

radius = prompt("Please enter the radius of the Circle!");

**function** AreaOfCircle(radius){

**var** area = Math.PI \* Math.pow(radius, 2);

**return** area;

}

console.log(AreaOfCircle(radius));

**Code 10: function.js**

## Array

**var** fastfood = ["Pizza", "Burger", "Sandwich"];

**var** drinks = [];

*/\*\**

*\* Finding the number of items in an array*

*\*/*

console.log(fastfood.length);

*/\*\**

*\* For every fast food we should have a drink*

*\*/*

**for**(**var** i = 0; i < fastfood.length; i++){

**switch**(i){

**case** 0:

drinks.push("lemonade");

**break**;

**case** 1:

drinks.push("coke");

**break**;

**case** 2:

drinks.push("sprite");

**break**;

**default**: console.log("No Drink Bro!");

}

}

*/\*\**

*\* It's in array format*

*\*/*

console.log(drinks);

*/\*\**

*\* For each to convert to String*

*\*/*

drinks.forEach(**function**($item, $index){

console.log($item);

});

**Code 11: array.js**

### Sort

This method sorts the items of an array.

The sort order can be either alphabetic or numeric, and either ascending (up) or descending (down).

By default, this method sorts the values as strings in alphabetical and ascending order.

**compareFunction(Optional)**

A function that defines an alternative sort order. The function should return a negative, zero, or positive value, depending on the arguments

function(a, b){return a-b}

When the sort() method compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value.

**function** init(){

**var** numbers = [60, 50, 62, 58, 54, 54];

numbers.sort(compareNumbers);

console.log(numbers);

}

**function** compareNumbers(num1, num2){

**if**(num1 > num2){

**return** 1;

}**else** **if**(num1 === num2){

**return** 0;

}**else**{

**return** -1;

}

}

window.onload = init;

**Code 12: sort.js**

## Simple Object

**var** fiat = {

make: "Fiat",

model: "500",

year: 1957,

color: "Blue",

passengers: 2,

convertible: **false**,

mileage: 88000,

drive: **function**(){

console.log(**this**.make + " Running: Zoom Zoom");

}

};

fiat.drive();

**Code 13: object.js**

|  |  |
| --- | --- |
| **Object provided by JavaScript** | **Objects provided by the web browser** |
| Date 🡪 Manipulate Date, Time | Document: Manipulate HTML |
| Math 🡪 Math Functions | Window: Browser Related Properties |
| RegExp 🡪 Patterns in Strings | Console: Debugging/Display Messages |
| JSON 🡪 Exchange JS Objects with apps |  |

## Document Object Model

### Get Element By ID

🡪document.getElementById(“ “)

🡪[document].innerHTML

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Basic File </**title**>

<**meta** charset = "UTF-8">

<**script** src = "document.js"></**script**>

</**head**>

<**body**>

<**h1** id = "text">What's up bro</**h1**>

<**button** onclick = "changeMe()">Change</**button**>

</**body**>

</**html**>

**function** changeMe(){

**var** text = document.getElementById("text");

text.innerHTML = "Nothing Much!";

}

**Code 14: document.js**

### Set Attribute

🡪document.setAttribute(“class”, “something”)

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Basic File </**title**>

<**meta** charset = "UTF-8">

<**script** src = "attribute.js"></**script**>

<**style**>**.modify**{**color**: rgba(255, 121, 123, 0.4);}</**style**>

</**head**>

<**body**>

<**h1** id = "text">What's up bro</**h1**>

<**button** onclick = "changeMe()">Change</**button**>

</**body**>

</**html**>

**function** changeMe(){

**var** text = document.getElementById("text");

text.innerHTML = "Nothing Much!";

text.setAttribute("class", "modify");

}

**Code 15: attribute.js**

To make sure the JavaScript is not running before the page loads we will write,

function init() {} then assign it to window.onload = init;

### Get Element By Tag Name

document.getElementsByTagName(“”)

This selects all the tags and will get back all elements of that tag name

### Get Element By Class Name

document.getElementsByClassName(“”)

Pass this method the name of a class, we will get back all elements that have that class, as a Node List

### Get Element By Name

document.getElementsByName(“”)

This method retrieves elements that have a name attribute with a value that matches the name we pass it

### Get Element By Query Selector

document.querySelector(“”)

This method takes a selector(just like a CSS selector) and returns the first element that matches.

### Get Element By Query Selector All

document.querySelectorAll(“”)

This method also takes a selector, but returns all the elements that match, as a Node List

## Primitive Types

### Undefined

It is similar to [isset() from PHP] where it **tests** whether to see if **a variable (or property, or array item) has been given a value.**

### Null

It is intended to represent an **object** that isn’t there. It may just mean that **object doesn’t exist** yet and needs to be created, or object doesn’t exist and we can skip it.

### NaN

**Not a Numbers, to represent numeric result**. Use the function isNan()

Ex: 0 divided by 0 will give NaN because this has no definite answer.

Ex: var b = “food” + 100

Ex: var c = Math.sqrt(-9)

## Strings

### Length

The length property holds the number of characters in the string. It’s quite handy for iterating through the characters of the string

var\_string.length

### Index Of

This method takes a string as an argument and returns the index of the first characters of the first occurrence of that argument in that string

var\_string.indexOf(string, starting\_point)

### Substring

Give the substring method two indices and it will extract and return the string contained within them.

var\_string.substring(starting\_point, ending\_point)

### Split

The split method takes a character that acts as a delimiter, and breaks the string into parts based on the delimiter.

var\_string.split(delimiter)

### Upper Case

Returns a string all lowercase character changed to uppercase characters

var\_string.toUpperCase()

### Trim

Removes all the white spaces in a string

var\_string.trim()

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Basic String Manipulation </**title**>

<**meta** charset = "UTF-8">

<**script**>

**function** init(){

*/\*\**

*\* Use Get Element By ID (Always to get value)*

*\*/*

**var** content = document.getElementById("patrick");

*/\*\**

*\* Use Text text content to retrieve the value*

*\*/*

**var** content = content.textContent;

*/\*\**

*\* Gives the length of the string*

*\*/*

**var** len = content.length;

*/\*\**

*\* Removes the white space*

*\*/*

**var** trimmed\_content = content.trim();

*/\*\**

*\* Upper Case All the string*

*\*/*

**var** upper\_content = trimmed\_content.toUpperCase();

console.log(content + ", length: " + len);

console.log("Trimmed Content: " + trimmed\_content + ", Upper Content: " + upper\_content);

*/\*\**

*\* Look for certain character*

*\*/*

**for**(**var** i = 0; i < len; i++){

**if**(content.charAt(i) == "S"){

console.log("S is for Superstar");

}

}

*/\*\**

*\* In which index does the star is found*

*\*/*

console.log(content.indexOf("Star"));

*/\*\* Gives the string at a given range*

*\*/*

**var** data = document.getElementById("data");

data = data.textContent;

**var** val = data.substring(6,9);

console.log(val);

*/\*\**

*\* Split the string*

*\*/*

data = data.split("|");

console.log(data);

combine = data.join(",")

console.log(combine);

}

window.onload = init;

</**script**>

</**head**>

<**body**>

<**h1** id = "patrick"> Patrick Star </**h1**>

<**p** id = "data">MySQL|PHP|JavaScript|HTML</**p**>

</**body**>

</**html**>

**Code 16: string.html**

### Lower Case

Returns a string all uppercase character changed to lowercase characters

var\_string.toLowerCase()

### Replace

Finds the substrings and replaces them with another string

var\_string.replace(delimiter, replace)

### Slice

Returns a new string that has part of the original string removed

var\_string.slice(start, end)

### Last Index Of

Just like indexOf(), but finds the last, not the first occurrence

lastIndexOf(string, start)

### Concatenation

Joins strings together

var\_string.concat(string1, string2, … )

### Match

Searches for matches in a string using regular expression. No Apostrophe inside the function. Returns null when no match found

var\_string.match(/string/modifier)

Regular Expression Modifiers

Modifiers can be used to perform case-insensitive more global searches:

|  |  |
| --- | --- |
| Modifier | Description |
| i | Perform case-insensitive matching |
| g | Perform a global match (find all matches rather than stopping after the first match) |
| m | Perform multiline matching |

Regular Expression Patterns

Brackets are used to find a range of characters:

|  |  |
| --- | --- |
| Expression | Description |
| [abc] | Find any of the characters between the brackets |
| [0-9] | Find any of the digits between the brackets |
| (x|y) | Find any of the alternatives separated with | |

Metacharacters are characters with a special meaning

|  |  |
| --- | --- |
| Metacharacter | Description |
| \d | Find a digit |
| \s | Find a whitespace character |
| \b | Find a match at the beginning or at the end of a word |

Quantifiers define quantities:

|  |  |
| --- | --- |
| Quantifier | Description |
| n+ | Matches any string that contains at least one n |
| n\* | Matches any string that contains zero or more occurrences of n |
| n? | Matches any string that contains zero or one occurrences of n |

Operators

|  |  |
| --- | --- |
| = | Assigns a value to a variable |
| == | Compares Object reference, Check values to see if equal |
| === | Compares values and types to see if equal |

<**script**>

**var** name = "Cupertino Spaniola";

*/\*\**

*\* Lower Case*

*\*/*

console.log(name.toLowerCase());

*/\*\**

*\* Replace*

*\*/*

console.log(name.replace("Span", "Ran"));

*/\*\**

*\* Slice*

*\*/*

console.log(name.slice(4, 6));

*/\*\**

*\* Match*

*\*/*

**var** name2 = "pan.peteroto";

**if**(name2.match(/oto/gi)){

console.log("Found it!");

}

**else**{

console.log("Where is he?");

}

*/\*\**

*\* Test*

*\*/*

**var** email = "peter.pan@gmail.com";

**var** regex = /[A-Za-z.\_]n?@[A-Za-z]n?.[a-z]n?[a-z]n?[a-z]n?/;

**if**(regex.test(email)){

console.log("Valid Email Address");

}

**else**{

console.log("Invalid Email Address");

}

</**script**>

**Code 17: string2.html**

## Handling Events

Whenever there’s an event, there is an opportunity for our code to handle it.

### On Click

The onclick event occurs when the user clicks on an element.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Event Handler</**title**>

<**meta** charset = "UTF-8"/>

<**script**>

**function** init(){

**var** event = document.getElementById("handler");

event.onclick = handleEventHandler;

**var** event\_form = document.getElementById("submit");

event\_form.onclick = handleEventForm;

}

**function** handleEventHandler(){

**var** handler = document.getElementById("handler");

handler.innerHTML = "Fire";

}

**function** handleEventForm(){

**var** fullname = document.getElementById("fullname");

**var** value = fullname.value;

console.log(value);

}

window.onload = init;

</**script**>

</**head**>

<**body**>

<**h1** id = "handler">Event Handler</**h1**>

<**input** type = "text" name = "text" id = "fullname"/><**br**/>

<**button** id = "submit">Confirm</**button**><**br**/><**br**/>

</**body**>

</**html**>

**Code 18: eventhandler.html**

### Handle Key Press

This event occurs when the user presses a key (on the keyboard).

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Handle Key Press</**title**>

<**meta** charset = "UTF-8">

<**script**>

**function** init(){

**var** element = document.getElementById("clickme");

element.onkeypress = handleevent;

}

**function** handleevent(){

element = document.getElementById("fire");

element.innerHTML = "Shoot";

}

window.onload = init;

</**script**>

</**head**>

<**body**>

<**input** type = "submit" id = "clickme" name = "submit"/>

<**h1** id = "fire">Fire</**h1**>

</**body**>

</**html**>

**Code 19: handlekeypress.html**

**Selecting Tags (Multiple)**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Tag Name</**title**>

<**meta** charset = "UTF-8">

</**head**>

<**script**>

**function** init(){

**var** superheroes = document.getElementsByTagName("p");

**for**(**var** i = 0; i < superheroes.length; i++){

superheroes[i].onclick = eventHandler;

}

}

**function** eventHandler(){

**var** text = document.getElementById("fire");

text.innerHTML = "Boom!";

}

window.onload = init;

</**script**>

<**body**>

<**h1** id = "fire"></**h1**>

<**p**> Superman </**p**>

<**p**> Batman </**p**>

<**p**> Spiderman </**p**>

</**body**>

</**html**>

**Code 20: tagname.html**

### Event Object

function eventHandler(eventObject){

var target = eventObject.target;

}

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Event Object</**title**>

<**meta** charset = "UTF-8"/>

<**script**>

**function** init(){

**var** words = document.getElementsByTagName("p");

**for**(**var** i = 0; i < words.length; i++){

words[i].onclick = eventHandler;

}

}

**function** eventHandler(eventObject){

**var** word = eventObject.target;

**var** id = word.id

**if**(word.hasAttribute("class")){

word.removeAttribute("class");

}

**else**{

word.setAttribute("class", "color");

}

console.log(id);

}

window.onload = init;

</**script**>

<**style**>

**.color**{

**color**: rgba(120, 120, 110, 0.7);

}

</**style**>

</**head**>

<**body**>

<**p** id = "zero">Zero</**p**>

<**p** id = "one">One</**p**>

<**p** id = "two">Two</**p**>

</**body**>

</**html**>

**Code 21: eventobject.html**

### Click

Get tis even when we click (or tap in the page)

### Load

The event we get when the browser has completed

### Mouse Move

When we move our mouse over an element

### Key Press

The event is generated every time we press a key

### Unload

This event is generated when we close the browser window, or navigate away from a web page.

### Mouse Over

When we put our mouse over an element, we will generate this event

### Mouse Out

We will generate this event when we move our mouse off an event

### Resize

When we resize our browser window, this event is generated

### Drag Start

If we drag an element in the page, we will generate this event

### Touch Start

On touch devices, we will generate a touch start event when we touch and hold an element

### Play

We will get this event when we will play <audio> or <video> play button

### Pause

We will get this event when we will pause <audio> or <video> play button

### Drop

We will get this event when we drop an element we have been dragging

### Touch End

We will get this even when we stop touching

### Set Time Out

setTimeout(function(){ alert("Hello"); }, 3000);

This setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds. Therefore, 1000 ms = 1 second. This function is only executed once.

Use the clearTimeout() method to prevent the function from running.

### Set Interval

This calls a function or evaluates an expression at specified intervals (in milliseconds).

This method will continue calling the function until clearInterval() is called, or the window is closed.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Local Time </**title**>

<**meta** charset = "UTF-8">

<**script**>

**function** init(){

setInterval(update,1000);

}

**function** update(){

**var** showtime = document.getElementById("date");

**var** date = **new** Date();

**var** localTime = date.toLocaleTimeString();

showtime.innerHTML = localTime;

}

window.onload = init;

</**script**>

<**style**>

</**style**>

</**head**>

<**body**>

<**p** id = "date"><**date**>Show time</**date**></**p**>

</**body**>

</**html**>

**Code 22: setinterval.html**

## First Class Functions

Start thinking about function as values, jut like numbers, strings, booleans, or objects. The ting that really makes a function value different from these other values is that we can invoke it.

**Steps**

* Assign the values to a variable (or store it in a data structure like an array or object)
* Pass the value to a function
* Return the value from a function

**First Class:** A value that can be treated like any other value in a programming language, including the ability to be assigned to a variable, passed as an argument, and returned from a function.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> First Class Functions </**title**>

<**meta** charset = "UTF-8">

<**script**>

**function** init(){

**var** telephone = callMe;

**var** answer = telephone("Peter");

console.log(answer);

}

**function** callMe(someone){

**return** (someone + " called me!");

}

window.onload = init;

</**script**>

</**head**>

<**body**>

</**body**>

</**html**>

**Code 23: firstclass.html**

### Nesting Function [Anonymous Function]

Used the anonymous function concept which is

setTimeout(function(){

alert(“Anonymous function!”);

},1000);

The precedence at which the compiler starts to read a code, the compiler reads the functions first then the variables.

**function** init(){

**var** fly = **function**(num){

**var** sound = "Flying Sound";

**function** wingFlapper(){

console.log(sound);

}

**for**(**var** i = 0; i < num; i++){

wingFlapper();

}

};

**function** quack(num){

**var** sound = "Quack";

**var** quacker = **function**(){

console.log(sound);

}

**for**(**var** i = 0; i < num; i++){

quacker();

}

}

quack(4);

fly(4);

}

window.onload = init;

**Code 24: nestinganonymous.js**

### Lexical Scope

Lexical just means we can determine the scope of a variable by reading the structure of the code, as opposed to waiting until the code runs to figure it out.

All local variables are stored in an environment. Java script functions are always evaluated in the same scope environment in which they were defined. Within a function, if we want to determine where a variable is coming from, search in its enclosing functions, from the most nested to the least.

### Closure

A closure is a function together with a referencing environment. If a variable in the function body isn’t defined locally, and it’s not a global, we can bet it’s from a function that it is nested in, and available in environment. A closure results when we combine a function that has free variables with an environment that provides variable bindings for all those free variables.

Therefore, for global variables we do not need to declare it is global inside the function because the variable is present in the environment.

**var** count = 0;

**function** init(){

console.log(counter());

console.log(counter());

console.log(counter());

}

**function** counter(){

count += 1;

**return** count;

}

window.onload = init;

**Code 25: closure.js**

Another way to create a closure is to pass a function to a function.

**function** init(){

makeTimer("Cooking Time Over!", 3000);

}

**function** makeTimer(message, time){

setTimeout(**function**(){alert(message)}, time);

}

window.onload = init();

**Code 26: closure2.js**

## Advanced Object Construction

Object constructors and functions are closely related. (Only in JavaScript)

**function** init(){

**var** fluffy = **new** Dog("Fluffy", "Poodle", 30);

**var** fido = **new** Dog("Fido", "Mixed", 38);

**var** spot = **new** Dog("Spot", "Chihuahua", 10);

**var** dogs = [fluffy, fido, spot];

dogs.forEach(**function**(dog){

console.log("Dog Name: " + dog.name);

console.log("Dog Breed: " + dog.breed);

console.log("Dog Weight: " + dog.weight);

console.log("==========================");

});

}

**function** Dog(name, breed, weight){

**this**.name = name;

**this**.breed = breed;

**this**.weight = weight;

}

window.onload = init;

**Code 26: basicobject.js**

Even constructed objects can have their own independent property

*/\*\* Insert Many type of same object into another \*/*

*/\*\**

*\**

*\*/*

**function** init(){

*/\*\**

*\* Created New Modded Dogs*

*\*/*

**var** fluffy\_mod = **new** Dog(fluffy);

**var** fido\_mod = **new** Dog(fido);

**var** spot\_mod = **new** Dog(spot);

*/\*\**

*\* Modded Dogs Array to see the results*

*\*/*

**var** mod\_dogs = [fluffy\_mod, fido\_mod, spot\_mod];

mod\_dogs.forEach(**function**(mod\_dog){

console.log("Dog Name: " + mod\_dog.name);

console.log("Dog Breed: " + mod\_dog.type);

console.log("Dog Weight: " + mod\_dog.weight);

console.log("Dog Running: " + mod\_dog.run);

console.log("==========================");

});

*/\*\**

*\* Constructed Objects with their individual property*

*\*/*

fido\_mod.owner = "Bob";

**delete** fido\_mod.weight;

**for**(**var** key **in** fido\_mod){

console.log(fido\_mod[key]);

}

}

*/\*\**

*\* Individual Dogs Objects*

*\*/*

**var** fluffy = {

name :"Fluffy",

type :"Poodle",

weight: 30

};

*/\*\**

*\* Individual Dogs Objects*

*\*/*

**var** fido = {

name :"Fido",

type :"Mixed",

weight: 38

};

*/\*\**

*\* Individual Dogs Objects*

*\*/*

**var** spot = {

name :"Spot",

type :"Chihuahua",

weight: 10

};

*/\*\**

*\* Dog Object (Inheritance)*

*\*/*

**function** Dog(param){

**this**.name = param.name;

**this**.type = param.type;

**this**.weight = param.weight;

**this**.run = **this**.name + " running ... running!";

}

window.onload = init;

**Code 26: objectliterals.js**

### Object Inside Object

**function** init(){

**var** trump = **new** HumanResource(**new** Teacher("Donald Trump", 4000));

**for**(**var** property **in** trump){

console.log(trump[property]);

}

}

**function** HumanResource(employee){

**this**.name = employee.name;

**this**.salary = employee.salary + 2000;

**this**.fire = "I can fire you!";

}

**function** Teacher(name, salary){

**this**.name = name;

**this**.salary = salary;

}

window.onload = init

**Code 27: inheritance.js**

### Date Objects

**function** init(){

*/\*\**

*\* Create Date Object*

*\*/*

**var** now = **new** Date();

console.log(now.toString());

console.log(now.getFullYear());

console.log(now.getDay());

}

window.onload = init;

**Code 27: others.js**

## Array Objects

How modulo division work

27 / 16 = 1, remainder 11

=> 27 mod 16 = 11

30 / 3 = 10, remainder 0

=> 30 mod 3 = 0

35 / 3 = 11, remainder 2

=> 35 mod 3 = 2

**[array].every()**

Every method takes a function and tests each value of the array to see if the function returns true or false when called on that value. If the function returns true for all the array items, then the result of every method is true.

**function** init(){

**var** array = **new** Array();

array.push(53, 234, 1234, 10, 1, 90 , 52, 92);

*/\*\**

*\* Normal Array*

*\*/*

console.log(array);

*/\*\**

*\* Reversed Array*

*\*/*

array.reverse();

console.log(array);

**var** findOdds = array.every(**function**(number){

**return** ((number%2) !== 0)

});

console.log(findOdds);

}

window.onload = init;

**Code 28: array.js**

## Prototypes

They are similar to abstract or interface. JavaScript object can inherit properties and behavior from other objects. JavaScript uses what is known as prototypal inheritance, and the object we are inheriting behavior from is called the prototype.

**function** init(){

**var** fido = **new** Dog("Fido", "Mixed", 38);

**var** fluffy = **new** Dog("Fluffy", "Poodle", 30);

**var** spot = **new** Dog("Spot", "Chihuahua", 10);

Dog.prototype.run = **function**(){

console.log("Run!");

};

console.log("Name: " + fido.name);

console.log("Breed: " + fido.breed);

console.log("Weight: " + fido.weight);

console.log("Run: " + fido.run);

}

**function** Dog(name, breed, weight){

**this**.name = name;

**this**.breed = breed;

**this**.weight = weight;

}

window.onload = init;

**Code 28: prototypes.js**

### Arguments

**function** printArgs(){

**for**(**var** i = 0; i < arguments.length; i++){

console.log(arguments[i]);

}

}

printArgs("sshh", "pssss", "isss");

**Code 28: arguments.js**

## Handling Operations

try{

}

catch(error){

console.log(“Error! ” + error.message);

}

## Add Event Listener

window.addEventListener(“load”, init, false)

window.onload = function(){

var div = document.getElementById(“clickme”)

div.addEventListener(“click”,handleClick, false);

};

function handleClick(e){

var target = e.target;

alert(“You clicked on “ + target.id);

target.removeEventListener(“click”, handleClick, false);

}

## Recursion

Recursion is based on mathematics 🡪 an algorithm to compute the Fibonacci number series. The Fibonacci number series is 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

function Fibonacci(n){

if((n === 0) || (n === 1)){

return 1;

}

else{

return (fibonacci(n-1) + fibonacci(n-2));

}

for(var i = 0; i < 10; i++){

console.log(“The Fibonacci of “ + i + “is “ + Fibonacci(i));

}

## Server-Side JavaScript

http.createServer(function(request, response){

response.writeHead(200, {“Content-Type”: “text/plain”});

response.write(“Hello World”);

response.end();

}).listen(8888);

## JavaScript Terms

### Window Object

window.innerWidth 🡪 Browser’s window width

window.innerHeight 🡪 Browser’s window height

window.close() 🡪 Closes the browser window

window.setTimeout() 🡪 Time out for window

window.setInterval() 🡪 Time interval for window

window.print() 🡪 Printing the page to the printer

window.confirm() 🡪 Similar to prompt, have okay or cancel button

window.history 🡪 Contains browsing history

window.location 🡪 URL of the current page, can be set to direct the browser to load a new page.

### Regular Expression Constructor

new RegExp(/^\d{3}-?\d{4}$/);

### JavaScript Object Notation (JSON)

var fidoString = ‘{“name”: “Fido”, “breed”: “Mixed”, “weight”: 38}’;

We are using single quotes around the JSON string. We have to use single quotes because the string contains double quotes, so JavaScript will get confused otherwise. This way, JavaScript knows this is one long string that contains other strings

var fido = JSON.parse(fidoString)

Passing the string to convert it to an Object

var fido = {name: “Fido”, breed: “Mixed”, weight: “38”}

var fidoString = JSON.stringify(fido);

Passing the object to convert it to a string

|  |  |
| --- | --- |
| **Debugging**  console.log()  alert()  prompt()  typeof  undefined  NULL  isNaN()  **Math**  Math.floor()  Math.PI  Math.random()  **Array**  [array].length()  [array].push()  [array].sort(*compareNumbers*)  [array].toString()  [array].reverse()  [array].join()  [array].forEach(function\_name)  **Events**  window.onload = init  window.onunload  window.resize  var = eventObject.target  var.id, var.src, var.hasAttribute()  var.removeAttribute(),  var.setAttribute()  [document].onclick  [document].onkeypress  [document].onmousemove  [document].onmouseover  [document].onmouseout  [document].pause  [document].onkeypress  [document].drop  [document].touchend  [document].pause  [document].play  [document].dragstart | **Timer**  setInterval()  setTimeout(function, time)  **Boolean False**  undefined  null  isNaN()  **String**  [string].indexOf(var, starting)  [string].split()  [string].trim()  [string].toUpperCase()  [string].toLowerCase()  [string].replace(delimiter, replace)  CharAt()  [string].substring(start, end)  [string].slice(start, end)  [regex[.test(string)  [string].concat(string1, string2,…)  **Date Object**  [new Date()].getFullYear()  [new Date()].getDay()  [new Dathe()].toString()  **HTML Document**  document.getElementById()  document.getElementsByTagName()  document.write()  [HTML Document].innerHTML  [HTML Document].textContent  [HTML Document].setAttribute()  [HTML Document].value  [HTML Document].getElementByClassName  [HTML Document].getElementByName  [HTML Document].querySelector  [HTML Document].querySelectorAll  [HTML Document].createElement(“”)  [HTML Document].appendChild()  **Array Object**  new Array()  [array].every() |

# JavaScript Intermediate

## AJAX PHP

**With Ajax**, the browser only sends and sends and receives the parts of a page that need to change. Sometimes the browser doesn’t have to talk to the server at all.

Ajax is a way of designing and building web pages that are as interactive and responsive as desktop applications. An **asynchronous request** is a request that occurs behind the scenes. Our users can keep working while the request is taking place.

* The browser can request multiple things from the server at the same time
* Browser requests return a lot faster
* Only the parts of the page that actually change are updated
* Server traffic is reduced
* the user can keep working while the page updates
* Some changes can be handled without a server round-trip

request.open(“GET”, url, true)

This method initializes the connection.

URL for the server-side script that will respond the request

true, it means that the request should by asynchronous. That is, the code in the browser should continue to execute while it’s waiting for the server to respond.

Asynchronous apps make requests using a JavaScript object, not a form submit.

**Some PHP Terminologies**

**basename(\_\_FILE\_\_, suffix)**

The basename() function returns the filename from a path.

\_\_FILE\_\_ is a magic function that returns the current file name

**http\_build\_query($associative\_array)**

Generate URL-encoded query string

**Some AJAX Terminologies**

**readyState**

A number that represents the current state of the request object

**status**

A status code returned by the server indicating, for example, success or that a requested resource is missing.

**responseXML**

It contains information sent back by the server in XML format

**responseText**

It contains textual information sent back by the server.

**statusText**

The status code message returned by the server, for example, “OK” for status 200

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>AJAX Concept</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "ajax.js"></**script**>

<**link** rel = "stylesheet" href = "ajax.css"/>

</**head**>

<**div** id = "thumbnail-pane">

<**img** src = "images/desktop.ico" class = "thumbnails" id = "desktop"/>

<**img** src = "images/harddrive.ico" class = "thumbnails" id = "harddrive"/>

<**img** src = "images/trash.ico" class = "thumbnails" id = "trash"/>

<**img** src = "images/folder.ico" class = "thumbnails" id = "folder"/>

</**div**>

<**div** id = "description-pane">

<**div** id = "description-text"></**div**>

</**div**>

</**html**>

**body**{

**padding**: 0px;

**margin**: 0px;

}

#description-pane{

**border**: 1px **solid** black;

**width**: 300px;

**height**: 200px;

}

#image-description{

**position**: **relative**;

**left**: 100px;

}

<?php

$data = ["desktop" => "Multi-tasking",

"harddrive" => "Saving Files",

"trash" => "Deleting Files",

"folder" => "Organizing Files"

];

**if**(isset($\_REQUEST)){

**echo** $data[$\_REQUEST["ImageID"]];

}

?>

**function** init(){

**var** thumbnails = document.getElementsByClassName("thumbnails");

**for**(**var** i = 0; i < thumbnails.length; i++){

thumbnails[i].onclick = thumbnailHandler;

}

}

**function** thumbnailHandler(event){

**var** target = event.target;

**switch**(target.id){

**case** "desktop":

**var** image = "images/" + target.id + ".ico";

createChild(image);

getDetails(target.id);

**break**;

**case** "harddrive":

**var** image = "images/" + target.id + ".ico";

createChild(image);

getDetails(target.id);

**break**;

**case** "trash":

**var** image = "images/" + target.id + ".ico";

createChild(image);

getDetails(target.id);

**break**;

**case** "folder":

**var** image = "images/" + target.id + ".ico";

createChild(image);

getDetails(target.id);

**break**;

**default**: "This is not a option!";

}

}

**function** createChild(image){

**var** elementExists = document.getElementById("image-description");

**var** description = document.getElementById("description-pane");

**if**(elementExists){

elementExists.remove();

}

**var** img = document.createElement("img");

img.setAttribute("id", "image-description");

img.src = image;

description.appendChild(img);

}

**function** getDetails(imageID){

**var** request = createRequest();

**if**(request == **null**){

alert("Unable to create request");

**return**;

}

**var** url = "ajax.php?ImageID=" + escape(imageID);

request.open("GET", url, **true**);

request.onreadystatechange = **function**(){

**if**(request.readyState == 4 || request.status == 200){

**var** text = document.getElementById("description-text");

text.innerHTML = request.responseText;

}

}

request.send(**null**);

}

**function** createRequest(){

**try**{

request = **new** XMLHttpRequest();

}

**catch**(tryMS){

**try**{

request = **new** ActiveXObject("Msxml2.XMLHTTP");

}**catch**(otherMS){

**try**{

request = **new** ActiveXObject("Microsoft.XMLHTTP");

}**catch**(err){

request = **null**;

}

}

}

**return** request;

}

window.onload = init;

**Code 1: ajax.js**

## Validation

**[element].onblur**

This event occurs when an object loses focus.

This event is most often used with form validation code (e.g. when the user leaves a form field).

The browse makes the server’s response available to our code through the properties of the request object. The browser sends our request object to the server, and the browser translates the server’s response.

**[element].disabled = true**

It disables the elements in the form

**[element].focus()**

It focuses the input form

**[element].select()**

It selects the input form

Refactoring code is pulling out common parts and putting those parts into a single, easily-maintainable function or method. Refactoring makes code easier to update and maintain.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Validation </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "validation.js"></**script**>

<**style**>

#passtooltip{

**display**: **none**;

}

#password-container:hover > #passtooltip{

**display**: **inline**-**block**;

**color**: rgba(0, 0, 255, 0.5);

}

**.con**{

**width**: 2%;

**height**: 2%;

}

</**style**>

</**head**>

<**body**>

<**form**>

<**label** for = "username"> Username </**label**><**br**/>

<**input** type = "text" id = "username"/>

<**img** id = "usernameImage" width = "1%" height = "1%"/>

<**span** id = "usernameCheck"></**span**>

<**br**/><**br**/>

<**div** id = "password-container">

<**label** for = "password"> Password </**label**><**br**/>

<**input** type = "password" id = "password"/>

<**span** id = "passtooltip">Use 6 characters minimum</**span**><**br**/><**br**/>

</**div**>

<**label** for = "password2"> Retype Password </**label**><**br**/>

<**input** type = "password" id = "password2"/>

<**img** id = "passwordImage" width = "1%" height = "1%"/>

<**span** id = "passwordCheck"></**span**>

<**br**/><**br**/>

<**input** type = "button" id = "register" value = "Register">

<**img** id = "submitImage" width = "1%" height = "1%"/>

<**span** id = "submitCheck"></**span**>

</**form**><**br**/><**br**/>

<**div** id = "image-container">

<**img** class = "con" src = "images/harddrive.ico" alt = "harddrive"/>

<**img** class = "con" src = "images/desktop.ico" alt = "desktop"/>

<**img** class = "con" src = "images/trash.ico" alt = "trash"/>

<**img** class = "con" src = "images/folder.ico" alt = "folder"/>

</**div**>

</**body**>

</**html**>

<?php

$names = ['Sam', 'Bob', 'Peter'];

**if**(isset($\_REQUEST)){

**if**(isset($\_REQUEST['username'])):

**if**(in\_array($\_REQUEST['username'], $names)):

**echo** "usernameexist";

**endif**;

**endif**;

**if**(isset($\_REQUEST['submit'])):

**echo** "Username: ".$\_REQUEST['user']. " Pass: ".$\_REQUEST['pass'];

**endif**;

}

?>

**var** checkStatus = [];

**var** position = 0;

**function** init(){

**var** username = document.getElementById("username");

username.onblur = checkUsername;

**var** password2 = document.getElementById("password2");

password2.onblur = checkPassword;

**var** register = document.getElementById("register");

register.addEventListener("click", checkSubmit, **false**);

setInterval(scrollImages, 2000);

}

**function** scrollImages(){

**var** images = document.querySelectorAll(".con");

console.log(images[position]);

**if**(position == 0){

**if**(images[images.length-1] == **undefined**){}

**else**{

images[images.length-1].removeAttribute("id");

}

}

**else** **if**(position < images.length){

images[position-1].removeAttribute("id");

}

images[position].setAttribute("id", "display");

**if**(position < (images.length - 1)){

position++;

}

**else** **if**(position = images.length){

position = 0;

}

}

**function** checkSubmit(){

**var** submitCheck = document.getElementById("submitCheck");

**var** submitImage = document.getElementById("submitImage");

**if**(checkStatus["username"] == "valid" &&

checkStatus["password"] == "valid"){

submitImage.src = "images/accept.png";

**var** request = createRequest();

**var** user = document.getElementById("username").value;

**var** pass = document.getElementById("password").value;

**var** url = "validation.php?submit=confirm&pass=" + pass

+ "&user=" + user;

request.open("GET", url, **true**);

request.onreadystatechange = **function**(){

**if**(request.status == 200 || request.readyState == 4){

console.log(request.responseText);

}

}

request.send(**null**);

}

**else**{

submitImage.src = "images/denied.png";

submitCheck.innerHTML = "Please, Fill the form";

}

}

**function** checkPassword(){

**var** password = document.getElementById("password");

**var** passCheck = document.getElementById("passwordCheck");

**var** passImage = document.getElementById("passwordImage");

**if**(escape(**this**.value.trim()) == escape(password.value.trim()) &&

escape(**this**.value.trim()) != "" &&

escape(password.value.trim())){

passCheck.innerHTML = "Password Matched";

document.getElementById("register").disabled = **false**;

passImage.src = "images/accept.png";

checkStatus['password'] = "valid";

}

**else**{

passCheck.innerHTML = "Password Do Not Match";

document.getElementById("register").disabled = **true**;

passImage.src = "images/denied.png";

checkStatus['password'] = "invalid";

}

}

**function** checkUsername(event){

**var** target = event.target;

**var** value = escape(target.value.trim());

**if**(value == **null** || value == ""){

**var** username = document.getElementById("usernameCheck");

**var** userimage = document.getElementById("usernameImage");

username.innerHTML = "You cannot have empty username";

userimage.src = "images/denied.png";

target.select();

target.focus();

}

**else**{

**var** request = createRequest();

**if**(request == **null**){

**return**;

}

**var** url = "validation.php?username=" + escape(target.value.trim());

request.open("GET", url, **true**);

request.onreadystatechange = usernameStatus;

request.send(**null**);

}

}

**function** usernameStatus(){

**if**(request.readyState == 4 || request.status == 200){

**var** username = document.getElementById("usernameCheck")

**var** userimage = document.getElementById("usernameImage");

**var** button = document.getElementById("register");

userimage.src = "images/loading.gif";

**if**(request.responseText == "usernameexist"){

username.innerHTML = "Username Exists";

userimage.src = "images/denied.png";

**var** user = document.getElementById("username");

user.focus();

user.select();

register.disabled = **true**;

checkStatus['username'] = "invalid";

}

**else**{

username.innerHTML = "Username Approved";

userimage.src = "images/accept.png";

checkStatus['username'] = "valid";

register.disabled = **false**;

}

}

}

**function** createRequest(){

**try**{

request = **new** XMLHttpRequest();

}**catch**(tryMS){

**try**{

request = **new** ActiveXObject("Msxml2.XMLHTTP");

}**catch**(otherMS){

**try**{

request = **new** ActiveXObject("Microsoft.XMLHTTP");

}**catch**(err){

request = **null**;

}

}

}

**return** request;

}

window.onload = init;

**Code 2: validation.js**

## Reaction

Event handlers are just properties. When we assign an event handler to an event on an HTML element, the handler becomes a property of the element, just like id or title.

var button = document.getElementById(“button”);

**button.onclick = eventHandler;**

Therefore, onclick became a property

A property can have only one value, if we assign a value to a property which is “eventHandler”, the property can only have a single value.

To have multiple event handlers with same property we need to use

**addEventListener().** We have been adding event handlers to elements to setting the event property directly. That’s called the **DOM Level 0 model.** It stands for Document Object Model Level 0

**DOM Level 2 model** allows us to give us a new method, called addEventListener()

**[element].addEventListener(“mouseover”, eventHandler, false);**

**[element].addEventListener(“mouseover”, evenetHandler, false);**

For Microsoft explorer we need to use,

**[element].addEvent(“onmouseover”, showHint);**

**function** init(){

**var** submit = document.getElementById("submit");

submit.addEventListener("click", showDisplay, **false**);

submit.addEventListener("click", showAnotherDisplay, **false**);

}

**function** showDisplay(event){

**var** target = event.target;

console.log(target.id);

}

**function** showAnotherDisplay(event){

**var** target = event.target;

console.log(target.value);

}

window.onload = init;

**Code 3: reaction.js**

Anytime we build cross-browser utility functions, store them methods in scripts that we can easily reuse in our other web applications.

**function** createRequest(){

**try**{

request = **new** XMLHttpRequest();

}

**catch**(tryMS){

**try**{

request = **new** ActiveXObject("Msxml2.XMLHttp");

}

**catch**(otherMS){

**try**{

request = **new** ActiveXObject("Microsoft.XMLHttp");

}

**catch**(err){

request = **null**;

}

}

}

**return** request;

}

**function** addEventListener(obj, eventName, handler){

**if**(document.attachEvent){

obj.attachEvent("on" + eventName, handler);

}**else** **if**(document.addEventListener){

obj.addEventListener(eventName, handler, **false**);

}

}

window.onload = init;

**Code 4: utils.js**

In **addEventListerner**, “this” refers to the owner of the executing function., an event is the owner of the handlers. In **attachEvent,** event handlers are owned by IE’s event framework. Not the active page object.

Event object has two properties **type and target**

Target 🡪 It is the object the event occurred on, like a tab or an image on a web page

Type 🡪 It is the name of the event that occurred. Ex: “onmouseover”

**function** init(){

**var** submit = document.getElementById("submit");

submit.addEventListener("click", eventHandler, **false**);

}

**function** eventHandler(event){

console.log(**this**.id);

console.log(event.target.id);

}

window.onload = init;

**Code 5: this.js**

## Family (Document Object Model)

### List of Useful DOM functions

≪Selectors≫

document.getElementById(“id”)

document.getElementsByClassName(“class-names”)

document.getElementsByTagNames(“tag-names”)

**document.getElementsByTagNames(“tag-names”)[0]**

Returns the first one

document.querySelector(“css-selectors”)

document.querySelectorAll(“css-selectors”)

**document.getElementById(“id”).getElementsTagName(“a”)**

Selecting document object using method chaining

**document.documentElement**

Grabs all <html> root element from an HTML document using the documentElement property 🡪 Get the root of the element

≪CHECK≫

[element].hasAttribute(“attr”)

≪CREATE≫

document.createElement(“element”)

document.createTextNode(“text”)

document.createAttribute(“id”, “ok”)

≪APPEND≫

[parentElement].appendChild([childElement])

[element].insertBefore(specificNode)

≪EDIT≫

[parentElement].replaceChild(“newChild”, “oldChild”)

≪DELETE≫

[element].removeAttribute(“id”)

[element].removeNode(“node”)

≪ROAMING≫

**firstChild, lastChild, nextSibling, parentNode, childNodes[node], previousSibling, nodeType, nodeName, nodeValue**

### Trees and Leaves

Set of related objects

p

a

p

h1

body

meta

title

head

html

**Fig 1: Trees and Leaves**

**Leaf:** A piece of markup that has no children such as an element with no text content, like <img> or textual data. Also know known as leaf data

**Node:** Any single piece of markup, such as an element or text. The <a> element is an element node, and its text is **text node**.

**Child:** Any piece of markup that is contained by another piece of markup.

**Branch:** A branch is a collection elements and content. So, the body branches are all the elements and text under the body element in the tree.

**Root Element:** The element in a document that contains all other elements. <html>

**Parent:** Any piece of markup that contains other markup. Parent Elements are also known as parent nodes.

### Family relationships Properties

div

**↑**

**table.parentNode**

**↑**

table

↓

**table.childNodes**

**↓ ↓ ↓ ↓**

tr tr tr tr

↓↓↓↓ ↓↓↓↓

thththth tdtdtdtd

**table.lastChild**

**table.firstChild.nextSibling**

**table.firstChild.firstChild**

**table.firstChild**

**Fig 2: Family Relationship**

There are inconsistencies in how browser treat whitespace. Never assume a browser will always a ignore, or always represent, whitespace.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Family Relationships</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "family.js"></**script**>

</**head**>

<**body**>

<**div** id = "container">

<**table** id = "employee" >

<**tr**>

<**th**>Name</**th**>

<**th**>Age</**th**>

<**th**>Occupation</**th**>

<**th**>Salary</**th**>

</**tr**>

<**tr**>

<**td**>Sam Randy</**td**>

<**td**>34</**td**>

<**td**>Pilot</**td**>

<**td**>35K</**td**>

</**tr**>

<**tr**>

<**td**>Michael Smith</**td**>

<**td**>23</**td**>

<**td**>Designer</**td**>

<**td**>29K</**td**>

</**tr**>

<**tr**>

<**td**>Allen Smith</**td**>

<**td**>59</**td**>

<**td**>Software Engineer</**td**>

<**td**>128K</**td**>

</**tr**>

<**tr**>

<**td**>Martha Curtis</**td**>

<**td**>39</**td**>

<**td**>Hotel Superviser</**td**>

<**td**>34K</**td**>

</**tr**>

</**table**>

</**div**>

</**body**>

</**html**>

**function** init(){

**var** employee = document.getElementById("employee");

console.log(employee.parentNode);

console.log(employee.childNodes);

console.log(employee.firstChild.nextSibling);

console.log(employee.firstChild.nextSibling.firstChild);

console.log(employee.firstChild.nextSibling.firstChild.nextSibling.nextSibling);

console.log(employee.firstChild.nextSibling.firstChild.nextSibling

.nextSibling.firstChild.nextSibling);

}

window.onload = init;

**Code 6: family.js**

### Node Name and Node Values

|  |  |  |  |
| --- | --- | --- | --- |
| **node** | **nodeType** | **nodeName** | **nodeValue** |
| document | document node | document | null |
| head | element node | head | null |
| p | element node | p | null |
| “Hello” | text node | text | “Hello” |
| id | attribute node | id | boom |

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Node </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "node.js"></**script**>

</**head**>

<**body**>

<**div** id = "check">Div Element Check</**div**>

</**body**>

</**html**>

**function** init(){

**var** x = document.getElementById("check").firstChild;

console.log(x.nodeName);

console.log(x.nodeType);

console.log(x.nodeValue);

}

window.onload = init;

**Code 6: node.js**

## AJAX XML

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

**<library>**

**<book>**

**<title>**Harry Potter**</title>**

**<author>**J.K Rowling**</author>**

**</book>**

**<book>**

**<title>** Da Vinci Code**</title>**

**<author>**Dan Brown**</author>**

**</book>**

**<book>**

**<title>** Oliver Twist**</title>**

**<author>** Charles Dickens**</author>**

**</book>**

**</library>**

<!DOCTYPE html>

<**html**>

<**head**>

<**head**>

<**title**>XML</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "library.js"></**script**>

</**head**>

</**head**>

<**body**>

<**div** id = "data"></**div**>

</**body**>

</**html**>

**function** init(){

**var** data = document.getElementById("data");

**var** request = **new** XMLHttpRequest();

url = "library.xml";

request.open("GET", url, **true**);

request.onreadystatechange = xmlHandler;

request.send(**null**);

}

**function** xmlHandler(event){

request = event.target;

**if**(request.readyState == 4 && request.status == 200){

**var** response = request.responseXML;

**var** div = document.getElementById("data");

**var** table = document.createElement("table");

**var** tr = document.createElement("tr");

**var** th = document.createElement("th");

**var** title = document.createTextNode("Title");

th.appendChild(title);

tr.appendChild(th);

**var** th = document.createElement("th");

**var** author = document.createTextNode("Author");

th.appendChild(author);

tr.appendChild(th);

table.appendChild(tr);

**var** books = response.getElementsByTagName("book");

**for**(**var** i = 0; i < books.length; i++){

*// console.log(books[i].firstChild.nextSibling.nodeName.trim());*

**var** tr = document.createElement("tr");

**var** td = document.createElement("td");

**var** title = document.createTextNode(books[i].firstChild.nextSibling.firstChild.nodeValue.trim());

td.appendChild(title);

tr.appendChild(td);

*// console.log(books[i].firstChild.nextSibling.nextSibling.nextSibling.nodeName.trim());*

**var** td = document.createElement("td");

**var** author = document.createTextNode(books[i].firstChild.nextSibling

.nextSibling.nextSibling

.firstChild.nodeValue.trim());

td.appendChild(author);

tr.appendChild(td);

table.appendChild(tr);

}

div.appendChild(table);

}

}

window.onload = init;

**Code 7: xml.js**

## AJAX CSV

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> CSV </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "csv.js"></**script**>

</**head**>

<**body**>

<**div** id = "data"></**div**>

</**body**>

</**html**>

**function** init(){

request = **new** XMLHttpRequest();

**var** url = "data.csv";

request.open("GET", url, **true**);

request.onreadystatechange = csvHandler;

request.send(**null**);

}

**function** csvHandler(){

**var** div = document.getElementById("data");

**if**(request.readyState == 4 && request.status == 200){

**var** data = request.response.split("-->");

**var** i = 0;

**while**(i < data.length){

**var** p = document.createElement("p");

**var** term = data[i] + ": " + data[i+1];

**var** text = document.createTextNode(term);

p.appendChild(text);

div.appendChild(p);

console.log(data[i]);

console.log(data[i+1]);

i += 2;

}

}

}

window.onload = init;

**Code 8: csv.js**

## AJAX JSON

We can convert JSON response in two ways

* Using **eval(‘(‘** **+** JSON string **+ ‘)’)**
* **JSON.parse(**JSON string**)**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>JSON</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "json.js"></**script**>

</**head**>

<**body**>

<**div** id = "data"></**div**>

</**body**>

</**html**>

<?php

$array = ["Sam" => "Gotta Go Fast", "Bob" => "Beaming through the light", "William" => "Do not need to"];

**if**(isset($\_REQUEST['call'])){

**echo** json\_encode($array);

}

?>

**function** init(){

request = createRequest();

**var** url = "json.php?call=calling";

request.open("GET", url, **true**);

request.onreadystatechange = jsonHandler;

request.send(**null**);

}

**function** jsonHandler(){

**if**(request.readyState == 4 && request.status == 200){

*// var json = eval('('+request.responseText +')');*

**var** json = JSON.parse(request.responseText);

console.log(json.Sam);

}

}

**function** createRequest(){

**try**{

request = **new** XMLHttpRequest();

}**catch**(tryMS){

**try**{

request = **new** ActiveXObject("mxxml2.XMLHTTP");

}**catch**(otherMS){

**try**{

request = **new** ActiveXObject("Microsoft.XMLHTTP")

}**catch**(err){

request = **null**;

}

}

}

**return** request;

}

window.onload = init;

**Code 9: json.js**

## Post

request = new XMLHttpRequest();

**var url = "data.php";**

**var requestData = "call=calling";**

**request.open("POST", url, true);**

In post method we are not going type in the arguments with the URL. We will separate out arguments and assign it to a variable

request.onreadystatechange = postHandler;

**request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");**

To receive the data from the server we need to send this header to get it in the form of array

**request.send(requestData);**

We need to send the arguments in this function.

|  |  |
| --- | --- |
| **Post** | **Get** |
| request = new XMLHttpRequest(); | request = new XMLHttpRequest(); |
| var url = **"data.php"**; | var url = "**data.php?call=calling**"; |
| var requestData = **"call=calling"**; | var requestData = **null**; |
| request.open(**"POST"**, url, true); | request.open(**"GET"**, url, true); |
| request.onreadystatechange = func | request.onreadystatechange = func |
| **request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");**  Only Required for Post | |
| request.send(**requestData**); | request.send(**null**)**;** |

<?php

$array = ["Sam" => "Gotta Go Fast", "Bob" => "Beaming through the light", "William" => "Do not need to"];

**if**(isset($\_REQUEST['call'])){

var\_dump($array);}

?>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Post</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "post.js"></**script**>

</**head**>

<**body**>

<**div** id = "data"></**div**>

</**body**>

</**html**>

**function** init(){

request = createRequest();

**var** url = "post.php";

**var** requestData = "call=calling";

request.open("POST", url, **true**);

request.onreadystatechange = postHandler;

request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

request.send(requestData);

}

**function** postHandler(){

**if**(request.readyState == 4 && request.status == 200){

console.log(request.responseText);

}

}

**function** createRequest(){

**try**{

request = **new** XMLHttpRequest();

}**catch**(tryMS){

**try**{

request = **new** ActiveXObject("mxxml2.XMLHTTP");

}**catch**(otherMS){

**try**{

request = **new** ActiveXObject("Microsoft.XMLHTTP")

}**catch**(err){

request = **null**;

}

}

}

**return** request;

}

window.onload = init;

**Code 9: post.js**

# JavaScript Advanced

## Document Object Model

**createElement(element)**

This method creates an Element Node with the specified name.

**createTextNode(element)**

This method creates a Text Node with the specified text.

**[element].appendChild(element)**

This method appends a node as the last child of a node.

var h = document.createElement("H1")  // Create a <h1> element  
var t = document.createTextNode("Hello World"); // Create a text node  
h.appendChild(t); // Append the text to <h1>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Add Songss </**title**>

<**meta** charser = "UTF-8"/>

<**script** src = "playlist.js"></**script**>

</**head**>

<**body**>

<**form**>

<**input** type = "text" id = "songTextInput" size = "40" placeholder = "Song Name">

<**input** type = "button" id = "addButton" value = "Add Song"/>

</**form**>

<**ul** id = "playlist">

</**ul**>

</**body**>

</**html**>

**function** init(){

**var** button = document.getElementById("addButton");

button.onclick = handleButtonClick;

}

**function** handleButtonClick(){

**var** textInput = document.getElementById("songTextInput");

**var** songName = textInput.value;

**var** list = document.createElement("li");

list.innerHTML = songName;

**var** unorderedList = document.getElementById("playlist");

unorderedList.appendChild(list);

}

window.onload = init;

**Code 1: playlist.js**

## Method Chaining

**movie.showtimes.length 🡪 Method Chaining**

**function** init(){

getShows(movie2);

}

**function** getShows(movie){

**for** (**var** i = 0; i < movie.showtimes.length; i++){

console.log("Next Show for " + movie.title + ": " +

movie.showtimes[i]);

}

}

**var** movie = {

title: "Plan 9 from Outer Space",

genre: "Cult Classic",

rating: 5,

showtimes: ["03:00 pm", "07:00 pm", "11:00 pm"]

};

**var** movie2 = {

title: "Forbidden Planet",

genre: "Classic Sci-Fi",

rating: 5,

showtimes: ["05:00 pm", "09:00 pm"]

};

window.onload = init;

**Code 2: methodchaining.js**

## Geolocation

Geolocation is not considered a first-class member of existing HTML5 standard, but that said, it is a standard of the W3C widely supported and pretty much everyone includes Geolocation in the list of important HTML5 APIs.

Geolocation API determines our location

**GPS**

Global Positioning System, supported by many newer mobile devices, provides extremely accurate location information based on satellites. Location data may include altitude, speed and heading information. To use it, device has to be able to see the sky, and it can take a long time to get a location. GPS can also be hard on our batteries.

**IP Address**

Location information based on our IP addresses uses an external database to map the IP address to a physical location. the advantage of this approach is that it can work anywhere, however, often IP addresses are resolved to locations such as ISP’s local office. This method as being reliable to the city or sometimes neighborhood level.

**Cell Phone**

Cell phone triangulation figures out our location based on our distance from one or more cell phone towers. More the towers, the more accurate our location will be. This method can be fairly accurate and works indoors (unlike GPS); it also can be much quicker than GPS. If we are in the middle of nowhere with only one cell tower, our accuracy is going to suffer.

**WIFI**

WIFI positioning uses one or more WIFI access points to triangulate our location. This method can be very accurate, works indoors and is fast. It requires us to be stationary

**navigator.geolocation**

This allows us to use the geographical position of a user

**navigator.geolocation.getCurrentPosition(displayLocation, displayerror)**

This method is used to return the user's position.

**displayLocation(position)**

position.coords.latitude;

position.coords.longitude;

Helps us to get the latitude and longitude

**displayError(error)**

**error.code 🡪 0 => Unknown error, 1 => User Permission Denied,**

**3 => Position Unavailable, 4 => Request Timed Out**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Geolocation </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "geolocation.js"></**script**>

</**head**>

<**body**>

<**div** id = "myLocation">

</**div**>

</**body**>

</**html**>

**function** getMyLocation(){

**if**(navigator.geolocation){

navigator.geolocation.getCurrentPosition(displayLocation, displayError);

}**else**{

console.log("Geolocation Not Supported");

}

}

**function** displayLocation(position){

**var** latitude = position.coords.latitude;

**var** longitude = position.coords.longitude;

**var** div = document.getElementById("myLocation");

div.innerHTML = "Latitude: " + latitude + "<br/>" +

"Longitude: " + longitude;

}

**function** displayError(error){

**var** errorTypes = {

0 : "Unknown error",

1 : "Permission denied by user",

2 : "Position is not available",

3 : "Request timed out"

};

**var** errorMessage = errorTypes[error.code];

**if**(error.code == 0 || error.code == 1){

errorMessage = errorMessage + " " + error.message;

}

**var** div = document.getElementById("muLocation");

div.innerHTML = errorMessage;

}

window.onload = getMyLocation;

**Code 3: geolocation.js**

### Computing Distance Between Two Geolocation Points

Therefore,

This uses the **‘haversine’ formula** to calculate the great-circle distance between two points – that is, the shortest distance over the earth’s surface – giving an ‘as-the-crow-flies’ distance between the points (ignoring any hills they fly over).

**Haversine Derivation**:

a = sin²(Δφ/2) + cos φ1 ⋅ cos φ2 ⋅ sin²(Δλ/2)

c = 2 ⋅ atan2( √a, √(1−a) )

**d = R ⋅ c**

where φ is latitude, λ is longitude, R is earth’s radius (mean radius = 6,371km);

note that angles need to be in radians to pass to trigonometric functions

var R = 6371e3; // metres

var φ1 = lat1.toRadians();

var φ2 = lat2.toRadians();

var Δφ = (lat2-lat1).toRadians();

var Δλ = (lon2-lon1).toRadians();

var a = Math.sin(Δφ/2) \* Math.sin(Δφ/2) +

Math.cos(φ1) \* Math.cos(φ2) \* Math.sin(Δλ/2) \* Math.sin(Δλ/2);

var c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

var d = R \* c;

**function** getGeo(){

**if**(navigator.geolocation){

navigator.geolocation.getCurrentPosition(displayLocation, displayError);

}**else**{

**var** div = document.getElementById("distance");

div.innerHTML = "Geolocation Not Supported!";

}

}

**function** displayLocation(pos){

**var** div = document.getElementById("distance");

latitude = pos.coords.latitude;

longitude = pos.coords.longitude;

div.innerHTML = "Longitude: " + latitude + "<br/>" +

"Longitude: " + longitude;

**var** lat2 = 47.624;

**var** long2 = -122.52088;

**var** distance = computingDistance(latitude, longitude, lat2, long2);

**var** computed = document.getElementById("computed");

computed.innerHTML = distance;

}

**function** displayError(err){

**var** errorTypes = {

0: "Unknown error",

1: "User Permission Denied",

2: "Position Unavailable",

3: "Request Timed Out",

};

**var** errorMessage = error.types[err.code];

**if**(err.code == 0 || err.code == 2){

errorMessage = errorMessage + " " + err.message;

}

**var** div = document.getElementById("distance");

div.innerHTML = errorMessage;

}

**function** computingDistance(lat1, long1, lat2, long2){

**var** radius = 6371;

**var** latitude1 = degreesToRadian(lat1);

**var** longitude1 = degreesToRadian(long1);

**var** latitude2 = degreesToRadian(lat2);

**var** longitude2 = degreesToRadian(long2);

**var** delta\_lat = degreesToRadian((lat2 - lat1));

**var** delta\_long = degreesToRadian((long2 - long1));

**var** a = Math.pow(Math.sin(delta\_lat/2), 2) +

Math.cos(latitude1/2) \* Math.cos(latitude2/2) \* Math.pow(Math.sin(delta\_long/2),2);

**var** c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

**var** d = radius \* c;

**return** d;

}

**function** degreesToRadian(deg){

**return** ((Math.PI \* deg)/180);

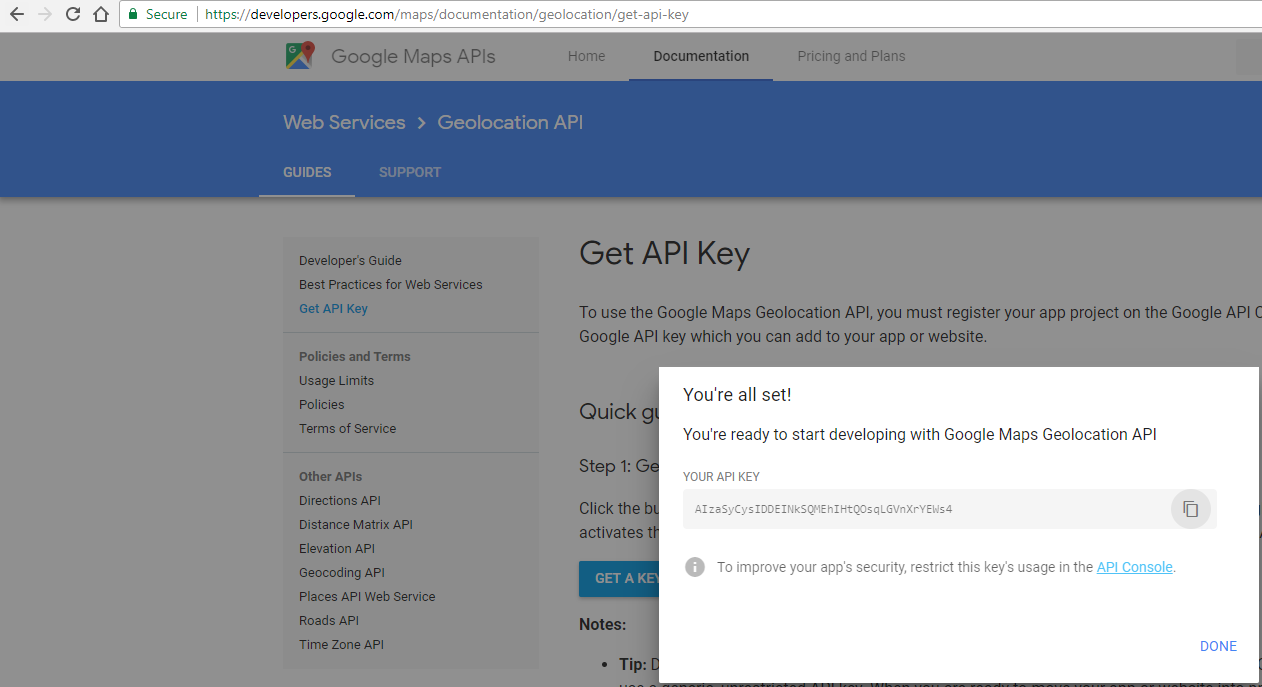
}

window.onload = getGeo;

**Code 4: computedDistance.js**

### Google Map

To use the google map, we need to generate a key and must have a Gmail account. Then generate a key for the google map from the website.

**Fig 1: googlegeneratekey.js**

Generate Key for the localhost

**AIzaSyD3sDQtedEfmXmqSmxL-3Fay7RM5NHWfPw**

Add this script

<script src=[https://maps.googleapis.com/maps/api/js?key= AIzaSyD3sDQtedEfmXmqSmxL-3Fay7RM5NHWfPw&callback=initMap](https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap) async defer></script>

For the map to appear in the html, we definitely to mention some of the properties in CSS

<**style**>

#map{**height**: 100%;}

**html**, **body** {**height**: 100%;**margin**: 0;**padding**: 0;}

</**style**>

Passing the Geolocation Coordination values in the Google Map function

google.maps.LatLng(latitude, longitude);

🡨 🡪 🡨 🡪

**google.maps** 🡺 It precedes all the methods of the google maps API.

**.LatLng** 🡺 This is the constructor, which takes out latitude and longitude, and returns a new object that holds them both.

**var** mapOptions = {

zoom: 19,

center: LatLng,

mapTypeId: google.maps.MapTypeId.ROADMAP

};

The **zoom** option can be specified 0 to 21. Bigger numbers correspond to being zoomed in more. 10 is about city sized.

The **center** option can be specified where the location can be, our location has been specified

The **mapTypeId** option can be SATELLITE or HYBRID etc. …

**var** divMap = document.getElementById("map");

map = **new** google.maps.Map(divMap, mapOptions);

Creating the **map** over here

**var** title = "Eastern";

**var** content = "Latitude: " + coords.latitude + ", Longitude: " + coords.longitude;

**var** marker = **new** google.maps.Marker({

position: LatLng,

map: map,

title: 'Eastern!',

clickable: **true**

});

Creates a **marker** over here, Clickable is set to true, because we want to be able to display an info window when it is clicked, window options let us see information when clicked.

**var** infoWindowOptions = {

content: content,

position: LatLng

};

**var** infoWindow = **new** google.maps.InfoWindow(infoWindowOptions);

google.maps.event.addListener(marker, "click", **function**(){

infoWindow.open(map);

});

}

<!DOCTYPE html>

<html>

<head>

<title>Google Map</title>

<meta charset = "UTF-8">

<style>

#map{height: 100%;}

html, body {height: 100%;margin: 0;padding: 0;}

</style>

<script src = "google.js"></script>

<script

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD3sDQtedEfmXmqSmxL-3Fay7RM5NHWfPw"

async defer>

</script>

</head>

<body>

<div id = "distance"></div>

<div id = "computed"></div>

<div id = "map"></div>

</body>

</html>

**function** getGeo(){

**if**(navigator.geolocation){

navigator.geolocation.getCurrentPosition(displayLocation, displayError);

}**else**{

**var** div = document.getElementById("distance");

div.innerHTML = "Geolocation Not Supported!";

}

}

**function** showMap(coords){

**var** LatLng = **new** google.maps.LatLng(coords.latitude, coords.longitude);

**var** mapOptions = {

zoom: 19,

center: LatLng,

mapTypeId: google.maps.MapTypeId.ROADMAP

};

**var** divMap = document.getElementById("map");

map = **new** google.maps.Map(divMap, mapOptions);

**var** title = "Eastern";

**var** content = "Latitude: " + coords.latitude + ", Longitude: " + coords.longitude;

**var** marker = **new** google.maps.Marker({

position: LatLng,

map: map,

title: 'Eastern!',

clickable: **true**

});

**var** infoWindowOptions = {

content: content,

position: LatLng

};

**var** infoWindow = **new** google.maps.InfoWindow(infoWindowOptions);

google.maps.event.addListener(marker, "click", **function**(){

infoWindow.open(map);

});

}

**function** displayLocation(pos){

**var** div = document.getElementById("distance");

latitude = pos.coords.latitude;

longitude = pos.coords.longitude;

div.innerHTML = "Longitude: " + latitude + "<br/>" +

"Longitude: " + longitude;

**var** lat2 = 47.624;

**var** long2 = -122.52088;

**var** distance = computingDistance(latitude, longitude, lat2, long2);

**var** computed = document.getElementById("computed");

computed.innerHTML = distance;

showMap(pos.coords);

}

**function** displayError(err){

**var** errorTypes = {

0: "Unknown error",

1: "User Permission Denied",

2: "Position Unavailable",

3: "Request Timed Out",

};

**var** errorMessage = error.types[err.code];

**if**(err.code == 0 || err.code == 2){

errorMessage = errorMessage + " " + err.message;

}

**var** div = document.getElementById("distance");

div.innerHTML = errorMessage;

}

**function** computingDistance(lat1, long1, lat2, long2){

**var** radius = 6371;

**var** latitude1 = degreesToRadian(lat1);

**var** longitude1 = degreesToRadian(long1);

**var** latitude2 = degreesToRadian(lat2);

**var** longitude2 = degreesToRadian(long2);

**var** delta\_lat = degreesToRadian((lat2 - lat1));

**var** delta\_long = degreesToRadian((long2 - long1));

**var** a = Math.pow(Math.sin(delta\_lat/2), 2) +

Math.cos(latitude1/2) \* Math.cos(latitude2/2) \* Math.pow(Math.sin(delta\_long/2),2);

**var** c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

**var** d = radius \* c;

**return** d;

}

**function** degreesToRadian(deg){

**return** ((Math.PI \* deg)/180);

}

window.onload = getGeo;

**Code 5: googlemap.js**

Using Google Map we can manipulate controls like pan control, Switch between Map and Satellite view and even the street view control. We can also use the direction services. Overlays can also be changed like heat map, check traffic, traffic overlay, custom overlays, our photos, custom markers.

### Tracking Movement

**getCurrentPosition(successHandler, errorHandler, positionOptions)**

The success handler (or callback) is called when a location is determined, and it is passed a position object.

The error handler is called when the browser can’t determine its location.

The position handler allows us to fine-tune the behavior of geolocation.

Coordinates

latitude

longitude

accuracy

altitude

altitudeAccuracy

heading

speed

Position

coords

timestamp

**Fig 2: Position 🡪 Coordinate (UML Diagram)**

Position Object contains the coords object(property) and the timestamp (property). Timestamp property contains the time the position object was created. This can be useful for knowing how old the location is.

Latitude, Longitude and accuracy are guaranteed to stay in the device. The rest may or may not be supported depending on our device.

**function** displayLocation(pos){

**var** div = document.getElementById("distance");

latitude = pos.coords.latitude;

longitude = pos.coords.longitude;

accuracy = pos.coords.accuracy;

div.innerHTML = "Longitude: " + latitude + "<br/>" +

"Longitude: " + longitude + "<br/>" +

"Accuracy: " + accuracy;

**var** lat2 = 47.624;

**var** long2 = -122.52088;

**var** distance = computingDistance(latitude, longitude, lat2, long2);

**var** computed = document.getElementById("computed");

computed.innerHTML = distance;

showMap(pos.coords);

}

**Code 6: updatedgooglemapwithaccuracy.js**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Tracking Movement </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "trackmovement.js"></**script**>

<**script**

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD3sDQtedEfmXmqSmxL-3Fay7RM5NHWfPw"

async defer>

</**script**>

<**style**>

#map{**height**: 50%;}

**html**, **body** {**height**: 100%;**margin**: 0;**padding**: 0;}

</**style**>

</**head**>

<**body**>

<**form**>

<**input** type = "button" id = "watch" value = "Watch me">

<**input** type = "button" id = "clearWatch" value = "Clear Watch">

</**form**>

<**div** id = "location"></**div**>

<**div** id = "map"></**div**>

</**body**>

</**html**>

**var** watchId = **null**;

**var** map;

**function** getMyLocation(){

**if**(navigator.geolocation){

**var** watchButton = document.getElementById("watch");

watchButton.onclick = watchLocation;

**var** clearWatchButton = document.getElementById("clearWatch");

clearWatchButton.onclick = clearWatch;

}

**else**{

console.log("Geolocation is not supported!");

}

}

**function** watchLocation(){

watchId = navigator.geolocation.watchPosition(displayLocation, displayError);

}

**function** clearWatch(){

**if**(watchId){

navigator.geolocation.clearWatch(watchId);

watchId = **null**;

}

}

**function** displayLocation(pos){

**var** div = document.getElementById("location");

latitude = pos.coords.latitude;

longitude = pos.coords.longitude;

accuracy = pos.coords.accuracy;

div.innerHTML = "Longitude: " + latitude + "<br/>" +

"Longitude: " + longitude + "<br/>" +

"Accuracy: " + accuracy;

**if**(map == **null**){

showMap(pos.coords);

}

}

**function** displayError(err){

**var** errorTypes = {

0: "Unknown error",

1: "User Permission Denied",

2: "Position Unavailable",

3: "Request Timed Out",

};

**var** errorMessage = error.types[err.code];

**if**(err.code == 0 || err.code == 2){

errorMessage = errorMessage + " " + err.message;

}

**var** div = document.getElementById("location");

div.innerHTML = errorMessage;

}

**function** showMap(coords){

**var** LatLng = **new** google.maps.LatLng(coords.latitude, coords.longitude);

**var** mapOptions = {

zoom: 19,

center: LatLng,

mapTypeId: google.maps.MapTypeId.ROADMAP

};

**var** divMap = document.getElementById("map");

map = **new** google.maps.Map(divMap, mapOptions);

**var** title = "Eastern";

**var** content = "Latitude: " + coords.latitude + ", Longitude: " + coords.longitude;

**var** marker = **new** google.maps.Marker({

position: LatLng,

map: map,

title: 'Eastern!',

clickable: **true**

});

**var** infoWindowOptions = {

content: content,

position: LatLng

};

**var** infoWindow = **new** google.maps.InfoWindow(infoWindowOptions);

google.maps.event.addListener(marker, "click", **function**(){

infoWindow.open(map);

});

}

window.onload = getMyLocation;

**Code 7: trackingmovements.js**

**watchPosition(successHandler, errorHandler)**

Our app calls the successHandler. watchPosition sits in the background and constantly monitors our positions. When our position changes, watchPosition calls our success handler function to report our new position. watchPosition continues to monitor our position (and report it to our success handler) until we clear it calling clearWatch

### Current Position with options

var positionOptions = {

enableHighAccuracy: false,

timeout: Infinity,

maximumAge: 0

}

**enableHighAccuracy:** We can control the accuracy of the geolocation

**timeout:** How long it gets to determine the user’s location

**maximumAgeL** How old the location can be. If the browser has a location that was determined sixty seconds go, and maximumAge is set to 90000 (90 seconds), then a call to getCurrentPosition would return the existing, cached position( the browser would not try to get a new one). But if the maximumAge was set to 30 seconds, the browser would be forced to determine a new position

## Asynchronous JavaScript and XML(AJAX)

Always check the size or length of the SQL object, it might hamper the when displaying with json\_encode()

Request object is created over here

**var request = new XMLHttpRequest();**

This sets up a request for us, using an HTTP GET request, which is the standard means of retrieving HTTP data. It also sets up the request to use the URL stored in our URL variable

**request.open(“GET”, url);**

200 is the browsers answer from the remote web service (when it is successful)

**request.onload** is same as **request.onreadystatechange** (request.onload is not supported by old browsers)

request.onload = function(){

if(request.status == 200){

//Do something

}

};

**request.sent(null);**

**JSON.stringify(array)** 🡪 turns array into an object 🡪 JSON Format

**JSON.parse(object)** 🡪 turns object into an array 🡪 JSON Format

**We check for request.status == 200 and request.readyState() == 4**

<!DOCTYPE html>

<html>

<head>

<title> AJAX </title>

<meta charset = "UTF-8"/>

<script src = "ajax.js"></script>

</head>

<body>

<ul id = "sales">

</ul>

</body>

</html>

<?php

$server\_name = "localhost";

$username = "root";

$pass = "";

$db = "riskyjobs";

$connection = mysqli\_connect($server\_name, $username, $pass, $db)

**or** **die** ("Server Denied");

$query = "SELECT \* FROM riskyjobs";

$data = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

$array\_data = **array**();

**while**($row = mysqli\_fetch\_array($data)){

$id = $row['job\_id'];

$title = $row['title'];

$city = $row['city'];

$state = $row['state'];

$zip = $row['zip'];

$company = $row['company'];

$date\_posted = $row['date\_posted'];

$array\_data[] = ["id" => "**$id**",

"title" => "**$title**",

"city" => "**$city**",

"state" => "**$state**",

"zip" => "**$zip**",

"company" => "**$company**",

"date\_posted" => "**$date\_posted**"

];

}

**echo**(json\_encode($array\_data));

?>

**function** init(){

**var** url = "ajax.php";

**var** request = **new** XMLHttpRequest();

request.open("GET", url);

*// request.onload = function(){*

request.onreadystatechange = **function**(){

**if**(request.status == "200" && request.readyState == "4"){

displayJSON(request.responseText);

}

};

request.send(**null**);

}

**function** displayJSON(data){

**var** dataObject = JSON.parse(data);

**var** ul = document.getElementById("sales");

**for**(**var** i = 0; i < dataObject.length; i++){

*// console.log("ID: " + dataObject[i].id);*

*// console.log("Title: " + dataObject[i].title);*

*// console.log("City: " + dataObject[i].city);*

*// console.log("State: " + dataObject[i].state);*

*// console.log("Zip: " + dataObject[i].zip);*

*// console.log("Company: " + dataObject[i].company);*

*// console.log("Date Posted: " + dataObject[i].date\_posted);*

*// console.log("==========================================");*

**var** li = document.createElement("li");

li.setAttribute("class", "title");

li.innerHTML = dataObject[i].title;

ul.appendChild(li);

}

}

window.onload = init;

**Code 8: ajax.js**

### JSON with Padding

We can’t retrieve data from a domain that is different from the domain the page itself was served from. Therefore, the **XMLHttpRequest** cross-origin request would fail. JSONP is a way to retrieve JSON objects by uing the <script> tag. It’s also a way of retrieving data and this avoids the same-origin security issues. **JSONP** stands for JavaScript Object Notation with Padding.

**Steps**

* We include a <script> element. The source for this script is the URL of a web service that is going to supply us with JSON data.
* The browser encounters the <script> element in the page and sends an HTTP request to the src URL
* **The server treats request and sends back the JSON string. Since the server sends back the JSON string, it first wraps it in a function call, like a call.**
* This time when the JSON response is parsed and interpreted, it is wrapped in a function call. And so that function is called, and the object created from the JSON string is passed to it.

[**http://localhost/ajax.php?callback=update**](http://localhost/ajax.php?callback=update)

### Cross Origin Resource Sharing

**header('content-type: application/json; charset=utf-8');**

**header("access-control-allow-origin: \*");**

To let everyone to see my data in JSON format, we need to flag to open. It is a specification that enables a truly open access across domain-boundaries. CORS defines how browsers and servers communicate when accessing sources across origins using HTTP headers to allow both the browser and the server to know enough about each other to determine if the request or response should succeed or fail.

<?php

$server\_name = "localhost";

$username = "root";

$pass = "";

$db = "riskyjobs";

$connection = mysqli\_connect($server\_name, $username, $pass, $db)

**or** **die** ("Server Denied");

$query = "SELECT \* FROM riskyjobs";

$data = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

$array\_data = **array**();

**while**($row = mysqli\_fetch\_array($data)){

$id = $row['job\_id'];

$title = $row['title'];

$city = $row['city'];

$state = $row['state'];

$zip = $row['zip'];

$company = $row['company'];

$date\_posted = $row['date\_posted'];

$array\_data[] = ["id" => "**$id**",

"title" => "**$title**",

"city" => "**$city**",

"state" => "**$state**",

"zip" => "**$zip**",

"company" => "**$company**",

"date\_posted" => "**$date\_posted**"

];

}

**echo** "update(".json\_encode($array\_data).")";

?>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> JSON P Basics </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "jsonp.js"></**script**>

</**head**>

<**body**>

<**h1**> Json P Data</**h1**>

<**ul** id = "data"></**ul**>

<**script** src = "jsonp.php?callback=update"></**script**>

</**body**>

</**html**>

window.onload = **function**(){};

**function** update(data){

console.log(data);

**var** ul = document.getElementById("data");

**for**(**var** i = 0; i < data.length; i++){

**var** li = document.createElement("li");

li.setAttribute("class", "title");

li.innerHTML = data[i].title;

ul.appendChild(li);

}

}

**Code 9: jsonp.js**

If we want to update the data after a certain interval of time without refreshing the page. Then we need to create the script tag manually through JavaScript.

**[parent\_element].replacechild(newChild, oldChild)**

This method replaces the children

**[element].remove()**

This method removes the element

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> JSON P Basics </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "jsonp.js"></**script**>

</**head**>

<**body**>

<**h1**> Json P Data</**h1**>

<**div** id = "info"></**div**>

</**body**>

</**html**>

<?php

$server\_name = "localhost";

$username = "root";

$pass = "";

$db = "riskyjobs";

$connection = mysqli\_connect($server\_name, $username, $pass, $db)

**or** **die** ("Server Denied");

$query = "SELECT \* FROM riskyjobs";

$data = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

$array\_data = **array**();

**while**($row = mysqli\_fetch\_array($data)){

$id = $row['job\_id'];

$title = $row['title'];

$city = $row['city'];

$state = $row['state'];

$zip = $row['zip'];

$company = $row['company'];

$date\_posted = $row['date\_posted'];

$array\_data[] = ["id" => "**$id**",

"title" => "**$title**",

"city" => "**$city**",

"state" => "**$state**",

"zip" => "**$zip**",

"company" => "**$company**",

"date\_posted" => "**$date\_posted**"

];

}

**echo** "update(".json\_encode($array\_data).")";

?>

window.onload = **function**(){

setInterval(handleRefresh, 3000);

};

**function** handleRefresh(){

**var** url = "jsonp.php?callback=update";

**var** newScript = document.createElement("script");

newScript.setAttribute("src", url);

newScript.setAttribute("id", "jsonp");

**var** oldScript = document.getElementById("jsonp");

**var** head = document.getElementsByTagName("head")[0];

**if**(oldScript == **null**){

head.appendChild(newScript);

}**else**{

head.replaceChild(newScript, oldScript);

}

}

**function** update(data){

console.log(data);

**var** div = document.getElementById("info");

**var** ulExists = document.getElementById("data");

**if**(ulExists){

ulExists.remove();

}

**var** ul = document.createElement("ul");

ul.setAttribute("id", "data");

**for**(**var** i = 0; i < data.length; i++){

**var** li = document.createElement("li");

li.setAttribute("class", "title");

li.innerHTML = data[i].title;

ul.appendChild(li);

}

div.appendChild(ul);

}

**Code 9: jsonpmodified.js**

We can also modify the URL by adding data and time to add uniqueness

**var url = “**[**http://localhost/jsonp.php?callback=update**](http://localhost/jsonp.php?callback=update)**” + “&random=” +**

**(newDate().getTime());**

## Canvas

**var selectedIndex = shape.selectedIndex;**

**var shape = shape[selectedIndex].value;**

Getting the selected value from the form

**var selection = document.getElementById(“mySelect”)**

**var option = document.createElement(“options”);**

**option.text = text;**

**option.value = value;**

**selection.options.add(option)**

Add Select options dynamically using javascript

**var context = canvas.getContext("2d");**

Needs 2D context from the canvas to draw

**context.fillStyle = “lightblue”**

fillStyle is a property, not a method. So, we set it, rather than call it. We can set it to a color by using the CSS color format which is color names (Ex: maroon, lightblue) or values like #cccff or rgb(0, 173, 239)

### Rectangle

**context.fillRect(initialPosX, initialPosY, width, height);**

The first two parameters are the x, y position of the rectangle

The last two parameters are the width and height of the rectangle

### Triangle

If we want to create a triangle, there is not create triangle method, therefore, we need to look into other canvas API built-in functions

**context.beginPath()**

Starting a new path

**context.moveTo(100, 50)**

This pin points the location where the line should start

**context.lineTo(250, 75)**

This method traces a path from the pencil’s current location to another point on the canvas.

Therefore, if the pencil’s initial point was {context.moveTo(100,50)} then it created a line between (100, 50) 🡪 (250, 75)

**Another context.lineTo(125, 30)** will create a line from (250, 75) 🡪 (125, 30)

Therefore, we have a line from (100, 50) 🡪 (250, 75) then another line from (250, 75) 🡪 (125, 30)

**context.closePath()**

This connects the starting point of the path to the last point in the current path

Therefore, (125, 30) 🡪 (250, 75)

### Circle

**context.beginPath()**

Starts a new path

**context.arc(intialPosX, initialPosY, radius, start∠, end∠, true)**

The first two parameters are the initial position

The third parameter is the radius

The fourth and fifth parameter is the starting and ending angle in radians

The fifth parameter is the direction whether it is clockwise (false), counterclockwise (true)

**context.fill()**

It is used to fill the color in the path

### Text

#### Alignment

**context.textAlign = “left”**

This property specifies where the anchor point for the text is. “start” is the default

Possible values are: start, end, left, right and center.

#### Fill and Stroke

**context.fillText(text, intialPositionX, initialPositionY, maxWidth)**

**context.strokeText(text, intialPositionX, initialPositionY, maxWidth)**

The first parameter is the text

The second and the third parameters are positions of the text

The fourth parameter is the size of the text

#### Font

**context.font = “2em Lucida Grande”**

**context.font = “italic bold 1.5em Times, serif”;**

**context.fillText(text, initialPositionX, initialPositionY)**

Adding fonts before adding the text.

#### BaseLine

The baseline property sets the alignment points in the font and determines the line our letters sit on.

**context.beginPath();**

**context.moveTo(100, 100);**

**context.lineTo(250, 100);**

**context.stroke();**

**context.textBaseline = “middle”;**

**context.fillText(“Alphabet”, 100, 100);**

Possible values are top, hanging, alphabetic, ideographic and bottom

#### Image

**var image = new Image();**

**image.src = "icon.ico";**

**context.drawImage(image, IntialPosX, InitialPosY, width, height);**

The first parameter is the image

The second and third parameters are the initial position

The fourth and fifth parameter are the width and the height

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Canvas </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "canvas.js"></**script**>

<**style**>

#drawing{

**border**: 1px **solid** black;

}

</**style**>

</**head**>

<**body**>

<**canvas** id = "drawing" width = "600" height = "200"></**canvas**>

<**form**>

<**label** for = "bgd\_color">Background Color:</**label**>

<**select** id = "bgd\_color">

<**option** value = "white" selected = "selected">White</**option**>

<**option** value = "black">Black</**option**>

</**select**>**&nbsp;**

<**label** for = "text\_color"> Text Color:</**label**>

<**select** id = "text\_color">

<**option** value = "white">White</**option**>

<**option** value = "black" selected = "selected">Black</**option**>

</**select**>**&nbsp;**

<**label** for = "shape"> Shape:</**label**>

<**select** id = "shape">

<**option** value = "square" selected = "selected">Square</**option**>

<**option** value = "circle">Circle</**option**>

</**select**><**br**/><**br**/>

<**label** for = "tweet">Tweet</**label**><**br**/>

<**input** type = "text" id = "tweet"/><**br**/><**br**/>

<**input** type = "button" id = "preview" value = "Preview"/>

</**form**>

</**body**>

</**html**>

**function** init(){

**var** previewButton = document.getElementById("preview");

previewButton.onclick = previewHandler;

}

**function** previewHandler(){

**var** canvas = document.getElementById("drawing");

**if**(canvas.getContext){

**var** context = canvas.getContext("2d");

**var** shape = document.getElementById("shape");

fillBackgroundColor(canvas, context);

**var** text\_color = document.getElementById("text\_color");

**var** selectedIndex = shape.selectedIndex;

**var** shape = shape[selectedIndex].value;

**var** selectedIndex = text\_color.selectedIndex;

**var** text\_color = text\_color[selectedIndex].value;

**if**(shape == "square"){

**for**(**var** i = 0; i < 50; i++){

drawSquare(canvas, context);

}

}

**if**(shape == "circle"){

**for**(**var** i = 0; i < 20; i++){

drawCircle(canvas, context);

}

}

drawText(canvas, context);

drawImage(canvas, context);

}

**else**{canvas.innerHTML = "Sorry, your browser do not support Canvas API";}

}

**function** fillBackgroundColor(canvas, context){

**var** bgd\_color = document.getElementById("bgd\_color");

**var** selectedIndex = bgd\_color.selectedIndex;

**var** bgd\_color = bgd\_color[selectedIndex].value;

context.fillStyle = bgd\_color;

context.fillRect(0, 0, canvas.width, canvas.height);

}

**function** drawSquare(canvas, context){

**var** locationX = Math.floor(Math.random() \* canvas.width);

**var** locationY = Math.floor(Math.random() \* canvas.height);

**var** size = Math.floor(Math.random() \* 40);

context.fillRect(locationX, locationY, size, size);

context.fillStyle = "lightblue";

}

**function** drawCircle(canvas, context){

context.fillStyle = "lightblue";

**var** locationX = Math.floor(Math.random() \* canvas.width);

**var** locationY = Math.floor(Math.random() \* canvas.height);

**var** size = Math.floor(Math.random() \* 40);

context.beginPath();

context.arc(locationX, locationY, size, 0, 2 \* Math.PI, **true**);

context.fillStyle = "lightblue";

context.fill();

}

**function** drawText(canvas, context){

**var** tweet = document.getElementById("tweet");

**var** tweet = tweet.value;

context.textAlign = "left";

context.font = "italic bold 1.4em Sem Times, serif"

context.fillStyle = "maroon";

context.fillText(tweet, 250, 100);

}

**function** drawImage(canvas, context){

**var** image = **new** Image();

image.src = "icon.ico";

context.drawImage(image, 20, 120, 70, 70);

}

window.onload = init;

**Code 10: canvas.js**

## Video

**<video controls autoplay**

**src = “location”**

**width = “480”**

**height = “360”**

**poster = “location”**

**id = “video”>**

**</video>**

Since the video element is a rich API, we can use in implement all kinds of interesting video behaviors and experiences.

**Call these methods**

* **play** 🡪 Play our video
* **pause** 🡪 Pauses the video
* **load** 🡪 Load the video
* **canPlayType** 🡪 It helps us determine which video types we can play programmatically

**Use these Properties**

* videoWidth
* videoHeight
* currentTime
* duration
* **ended**
* error
* **loop**
* **muted**
* paused
* readyState
* seeking
* volume

Some of them can be set (like loop and muted)

Some of them are read only (like currentTime and error)

**Catch these Events**

* play
* pause
* progress
* error
* timeupdate
* ended
* abort
* waiting
* loadeddata
* loadedmetadata
* volumechange

These are all the event handlers

### Alternative Event Handler

**video.onended = nextVideo**

**video.addEventListener(“ended”, nextVideo, false)**

First parameter is the event that we are listening for. We do not put “on” before the event name

Second parameter is the function that we are going to call

Third parameter controls some advanced methods of getting events if it is set to true

### Video Processing

* The video player decodes and plays the video behind the scenes
* Video copied frame by frame into a (hidden) buffer canvas and processed
* After a frame is processed, it is copied to another canvas to be viewed.

Processing Video Using Scratch Buffer

It is a proven technique for mini9mizing visual glitches during intensive video and image processing, it’s known as using a “scratch buffer”. By processing a frame of video in a buffer and then copying it all in one fell swoop to the display canvas, we minimize visual issues.

* The browser decodes the video into a series of frames. Each frame is a rectangle of pixels with a snapshot of the video at a given point in time.
* As each frame is decoded we copy it into the canvas that is acting as a scratch buffer.
* We iterate over scratch buffer, pixel by pixel, passing each pixel to a function for processing
* After all the pixels in the scratch buffer are processed, we copy them from the scratch buffer canvas to the display canvas.
* Then we repeat the process on every frame as it is decoded by the video object

Creating Buffer

**buffer.drawImage(video, 0, 0, bufferCanvas.width, bufferCanvas.height);**

It takes an image and draws that image onto the canvas, at an x, y position for a given width and height.

The time we are getting an image from the video. By specifying the video as the source, drawImage gets one frame of the video as image data

**var frame = buffer.getImageData(0, 0, bufferCanvas.width, bufferCanvas.height);**

Then we grab the image data from the canvas context and store it in a variable, frame, so we can process it.

**var length = frmae.data.length/4;**

One of the property of frame is frame.data, and length is a property of frame.data. The length is actually four times longer than the size of the canvas because each pixel has four values: RGBA

**var length = frame.data.length/4;**

**for(var i = 0; i < length; i++){**

**var r = frame.data[i\*4 + 0];**

**var g = frame.data[i\*4 + 1];**

**var b = frame.data[i\*4 + 2];**

**if(effectFunction){**

**effectFunction(i, r, g, b, frame.data);**

**}**

**}**

Now we loop over the data and get the RGB values for each pixel. Each pixel takes up four spaces in the array, so we grab r from the first position, g from the second and b from the third

**display.putImageData(frame, 0, 0);**

The frame data has been processed, so we use the context put **ImageData** method to put the data into the display canvas. This method takes the data in frame and writes it into the canvas at the specified x, y position.

### List of Effects

Black and White

**if**(effectFunction == "noir"){

**var** brightness = (3\*r + 4\*g +b) >>> 3;

**if**(brightness < 0){brightness = 0;};

frame.data[i\*4 + 0] = brightness;

frame.data[i\*4 + 1] = brightness;

frame.data[i\*4 + 2] = brightness;

}

Sepia

**if**(effectFunction == "western"){

**var** brightness = (3\*r + 4\*g +b) >>> 3;

frame.data[i\*4 + 0] = brightness + 40;

frame.data[i\*4 + 1] = brightness + 20;

frame.data[i\*4 + 2] = brightness + 20;

}

Negative

**if**(effectFunction == "scifi"){

**var** offset = i \* 4;

frame.data[offset] = Math.round(255 - r);

frame.data[offset+1] = Math.round(255 - g);

frame.data[offset+2] = Math.round(255 - b);

}

Cartoon

**if**(effectFunction == "cartoon"){

**var** offset = i \* 4;

**if**(frame.data[offset] < 120){

frame.data[offset] = 80;

frame.data[offset+1] = 80;

frame.data[offset+2] = 80;

}

**else**{

frame.data[offset] = 255;

frame.data[offset+1] = 255;

frame.data[offset+2] = 255;

}

}

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Video</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "video.js"></**script**>

</**head**>

<**body**>

<**video** width = "240" height = "180" poster = "icon.ico" id = "video"></**video**>

<**canvas** id = "buffer" width = "240" width = "180"></**canvas**>

<**canvas** id = "display" width = "240" width = "180"></**canvas**>

<**div** id = "controls">

<**form**>

<**input** type = "button" id = "play"

class = "controls" value = "Play"/>**&nbsp;**

<**input** type = "button" id = "pause"

class = "controls" value = "Pause"/>**&nbsp;**

<**input** type = "button" id = "loop"

class = "controls" value = "Loop"/>**&nbsp;**

<**input** type = "button" id = "mute"

class = "controls" value = "Mute"/>

</**form**>

</**div**><**br**/><**br**/>

<**div** id = "effects">

<**form**>

<**input** type = "button" id = "normal"

class = "effects" value = "Normal"/>**&nbsp;**

<**input** type = "button" id = "noir"

class = "effects" value = "Noir"/>**&nbsp;**

<**input** type = "button" id = "western"

class = "effects" value = "Western"/>**&nbsp;**

<**input** type = "button" id = "scifi"

class = "effects" value = "Sci-Fi"/>**&nbsp;**

<**input** type = "button" id = "cartoon"

class = "effects" value = "Cartoon"/>**&nbsp;**

</**form**>

</**div**><**br**/><**br**/>

<**div** id = "videoSelection">

<**form**>

<**input** type = "button" id = "video1"

class = "videoSelection" value = "Video 1"/>**&nbsp;**

<**input** type = "button" id = "video2"

class = "videoSelection" value = "Video 2"/>**&nbsp;**

</**form**>

</**div**>

</**body**>

</**html**>

**var** position = 0;

**var** playlist;

**var** video;

**var** effectFunction = **null**;

**function** init(){

playlist = ['magic', 'urma'];

video = document.getElementById("video");

video.addEventListener("ended", nextVideo, **false**);

video.addEventListener("play", processFrame, **false**);

**var** controls = document.querySelectorAll("input.controls");

**for**(**var** i = 0; i < controls.length; i++){

controls[i].onclick = handleControl;

}

**var** effects = document.querySelectorAll("input.effects");

**for**(**var** i = 0; i < effects.length; i++){

effects[i].onclick = handleEffect;

}

**var** videoSelection = document.querySelectorAll("input.videoSelection");

**for**(**var** i = 0; i < videoSelection.length; i++){

videoSelection[i].onclick = handleVideoSelection;

}

video.src = playlist[position] + getFormatExtension();

video.load();

video.play();

}

**function** nextVideo(){

position++;

**if**(position >= position.length){

postion = 0;

}

video.src = playlist[position] + getFormatExtension();

video.load();

video.play();

}

**function** getFormatExtension(){

**if**(video.canPlayType("video/mp4") != ""){

**return** ".mp4";

}

**else**{

**return** ", This video is not supported!";

}

}

**function** handleControl(event){

**var** target = event.target;

**switch**(target.id){

**case** "play":

video.play();

**break**;

**case** "pause":

video.pause();

**break**;

**case** "loop":

**if**(video.loop == **false**){

video.loop = **true**;

}**else**{

video.loop = **false**;

}

**break**;

**case** "mute":

**if**(video.muted == **false**){

video.muted = **true**;

}**else**{

video.muted = **false**;

}

**break**;

**default**: "Sorry this is not an option";

}

}

**function** handleVideoSelection(event){

**var** target = event.target;

**if**(target.id == "video1"){

position = 0;

video.src = playlist[position] + getFormatExtension();

video.load();

video.play();

}

**if**(target.id == "video2"){

position = 1;

video.src = playlist[position] + getFormatExtension();

video.load();

video.play();

}

}

**function** handleEffect(event){

**var** target = event.target;

**switch**(target.id){

**case** "normal":

effectFunction = **null**;

**break**;

**case** "noir":

effectFunction = "noir";

**break**;

**case** "western":

effectFunction = "western";

**break**;

**case** "scifi":

effectFunction = "scifi";

**break**;

**case** "cartoon":

effectFunction = "cartoon";

**break**;

**default**: "Not a valid effect";

}

}

**function** processFrame(){

**if**(video.paused || video.ended){

**return**;

}

**var** bufferCanvas = document.getElementById("buffer");

**var** buffer = bufferCanvas.getContext("2d");

**var** displayCanvas = document.getElementById("display");

**var** display = displayCanvas.getContext("2d");

buffer.drawImage(video, 0, 0, bufferCanvas.width, bufferCanvas.height);

**var** frame = buffer.getImageData(0, 0, bufferCanvas.width, bufferCanvas.height);

**var** length = frame.data.length/4;

**for**(**var** i = 0; i < length; i++){

**var** r = frame.data[i\*4 + 0];

**var** g = frame.data[i\*4 + 1];

**var** b = frame.data[i\*4 + 2];

**if**(effectFunction == "noir"){

**var** brightness = (3\*r + 4\*g +b) >>> 3;

**if**(brightness < 0){brightness = 0;};

frame.data[i\*4 + 0] = brightness;

frame.data[i\*4 + 1] = brightness;

frame.data[i\*4 + 2] = brightness;

}

**if**(effectFunction == "western"){

**var** brightness = (3\*r + 4\*g +b) >>> 3;

frame.data[i\*4 + 0] = brightness + 40;

frame.data[i\*4 + 1] = brightness + 20;

frame.data[i\*4 + 2] = brightness + 20;

}

**if**(effectFunction == "scifi"){

**var** offset = i \* 4;

frame.data[offset] = Math.round(255 - r);

frame.data[offset+1] = Math.round(255 - g);

frame.data[offset+2] = Math.round(255 - b);

}

**if**(effectFunction == "cartoon"){

**var** offset = i \* 4;

**if**(frame.data[offset] < 120){

frame.data[offset] = 80;

frame.data[offset+1] = 80;

frame.data[offset+2] = 80;

}

**else**{

frame.data[offset] = 255;

frame.data[offset+1] = 255;

frame.data[offset+2] = 255;

}

}

}

display.putImageData(frame, 0, 0);

setTimeout(processFrame, 0);

}

window.onload = init;

**Code 11: video.js**

### Error Events

**MEDIA\_ERR\_ABORTED=1**

It is used any time the process of getting the video over the network is aborted by the browser (possibly at a user’s request)

**MEDIA\_ERR\_NETWORK=2**

It is used whenever a network retrieval of the video is interrupted by a network error.

**MEDIA\_ERR\_DECODE=3**

It is used whenever the decoding of a video fails. This could happen because the encoding uses features the browser can’t support or because the file is corrupt

**MEDIA\_ERR\_SRC\_NOT\_SUPPORTED=4**

It is used when the specified video source cannot be supported because of a bad URL or because the source type isn’t decodable by the browser.

**video.addEventListener(“error”, errorHandler, false)**

**{video.error.code}**

This handles the error code

## Web Storage

It is a simple JavaScript API in the browser for storing key/value pairs that are persistent. All browsers provide 5 to 10 megabytes of storage in every user’s browser.

**localStorage.setItem(“key”, “value”);**

The Web Storage API is available to us through the localStorage object. It saves the data in key and value style

**localStorage.getItem(“key”);**

**localStorage["key"]**

To get the item

When we save integer in this method, it stores in the form of string. Therefore, to parse this we need to use additional steps

localStorage.setItem("boom", "5");

**var** boom = 5 + parseInt(localStorage.getItem("boom"));

console.log(boom);

Local storage also gives: a property (length) and a method, key

**localStorage.length**

It tells us how many items are there in the storage

**localStorage.key(numeric)**

It also saves in the numeric order

for(var i = 0; i < localStorage.length; i++){

var key = localStorage.key(i);

console.log(localStorage[key]);

}

**localStorage.removeItem(key)**

Removes the local storage key-value pair

|  |  |
| --- | --- |
| **Cookies** | **Local Storage** |
| 4kB Size Limit | 5 to 10 MB Size Limit |
| Can store numeric, string | Numeric are converted to strings |

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Web Storage </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "webstorage.js"></**script**>

</**head**>

<**body**>

</**body**>

</**html**>

**function** init(){

localStorage.setItem("sticky\_0", "Pick up dry cleaning");

localStorage.setItem("sticky\_1", "Go to the bookstore");

**for**(**var** i = 0; i < localStorage.length; i++){

**var** key = localStorage.key(i);

console.log(localStorage[key]);

}

}

window.onload = init;

**Code 12: sticky.js**

## Web Workers

Single-threaded mode of computing by JavaScript works great but it starts to impact when it is computationally intensive. Before HTML 5, one thread used to control the pages and apps but with Web Workers we have not got a way to create another thread of control to help out.

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished. A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

* One of the requirements of a worker is that it should not have access to the DOM (or to any state of the main browser thread for that matter). Web Workers designers chose to pass a JavaScript URL.
* When we send an object, the worker gets a copy of it. any changes the worker makes will not affect the object in our main page. The worker is executing in a different environment than our main page. so, we have no access to objects there. The same is true for objects the worker sends us; we get a copy of them.
* Workers can access localStorage and make XMLHttpRequest

|  |  |
| --- | --- |
| **Manager** | **Worker** |
| **var worker = new Worker(worker.js)**  Defining a new worker  **worker.postMessage(message)**  Sends message to the worker  **worker.terminate()**  Terminate a worker  **worker.onerror = function(error){**   * + **error.filename**   + **error.lineno**   + **error.message**   **}**  Handling errors in codes | **importScripts(“another\_helper.js”)**  Get help from another JavaScript file  **postMessage(response)**  Give response  **setInterval(function, time)**  Repeat a function in worker  **var worker = new Worker(“subworker.js”)**  Create a sub-worker |

### Basic Structure

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Web Worker</**title**>

<**meta** charser = "UTF-8">

<**script** src = "manager.js"></**script**>

</**head**>

<**body**>

<**p** id = "output"></**p**>

</**body**>

</**html**>

**Code 13: webworker.html**

**function** init(){

**var** worker = **new** Worker('worker.js');

worker.postMessage("ping");

worker.onmessage = **function**(event){

message = event.data;

worker = event.target;

document.getElementById("output").innerHTML = worker + ": " + message;

console.log(message);

};

}

window.onload = init;

**Code 14: manager.js**

onmessage = **function**(event){

**if**(event.data == "ping"){

importScripts("another\_helper.js");

postMessage(message);

}

}

**Code 15: worker.js**

var message = "I am helping too!";

**Code 16: another\_helper.js**

## Modernizr

There is no uniform way to detect browser support for an API, there is no uniform way of doing so. Modernizr is an open source JavaScript library that provides a uniform interface for detecting browser support. Mordenizer takes care of all the details of the different means of detection, even factoring in all the edge cases around older browsers.

To detect support

**if(Modernizr.geolocation){}**

**if(Modernizr.localStorage){}**

**if(Mordernizr.video){}**

## Offline Web Apps

We create a cache manifest file that contains a list of all the files our app needs to work and the browser will download all those files, and switch to the local files if and when our device goes offline. To tell our web page that it has a manifest file, we simply add the filename of the cache manifest file to our <html> tag

This must be added to the html tag

<html manifest = “notetoself.manifest”>

notetomyself.manifest file will contain

CACHE MANIFEST 🡪 Every cache manifest file must start with this

CACHE:

notetomyself.html

notetomyself.css

notetomyself.js

We can also add two other sections to the file,

FALLBACK 🡪 It specifies what file to use if we try to access a file that isn’t cached, and

NETWORK 🡪 It specifies that should never be cached (for example, visit tracking resources)

In our Apache Server 🡪 AddType text/cache-manifest .manifest 🡪 needs to be present

To notify the events

**window.applicationCache.addEventListener**(“error”, errorHandler, false);

## Web Sockets

With respect to JSONP, XMLHttpRequest, there is another way to of communicating which web sockets. In all the cases we used a request/response model based on HTTP

Web Socket is a new API that allows us to keep an open connection with a web service so that any time new data is available the service can just sent id us.

var socket = new WebSocket(<https://sphotonkhan.com>);

sockent.onopen = function(){console.log(“Socket Open”);

socket.postMessage(“Boom”)

socket.onmessage = function(event){console.log(event.data)}};

## JavaScript Terms

≪Document Object Model≫

**createElement(element)**

This method creates an Element Node with the specified name.

**createTextNode(element)**

This method creates a Text Node with the specified text.

**[element].appendChild(element)**

This method appends a node as the last child of a node.

**[parent\_element].replacechild(newChild, oldChild)**

This method replaces the children.

**[element].remove()**

This method removes the element

**[element].parentNode()**

This method allows to go the parent node

≪Geolocation≫

**navigator.geolocation**

This allows us to use the geographical position of a user

**navigator.geolocation.getCurrentPosition(displayLocation, displayerror)**

This method is used to return the user's position.

**getCurrentPosition(successHandler, errorHandler, positionOptions)**

The success handler (or callback) is called when a location is determined, and it is passed a position object. The error handler is called when the browser can’t determine its location.

The position handler allows us to fine-tune the behavior of geolocation

**displayLocation(position)**

position.coords.latitude;

position.coords.longitude;

position.coords.accuracy;

**watchPosition(successHandler, errorHandler)**

Our app calls the successHandler. watchPosition sits in the background and constantly monitors our positions. When our position changes, watchPosition calls our success handler function ro report our new position. watchPosition continues to monitor our position (and report it to our success handler) until we clear it calling clearWatch

≪Google Map≫

**new google.maps.Map(document, mapOptions)**

This creates and returns map object

**new google.maps.LatLng(latitude, longitude);**

google.maps 🡺 It precedes all the methods of the google maps API.

.LatLng 🡺 This is the constructor, which takes out latitude and longitude, and returns a new object that holds them both.

**new google.maps.Marker(markerOptions)**

This constructor creates marker object

**new google.maps.InfoWindow(infoWindowOptions)**

This constructor creates Info Window

**google.maps.event.addListener(marker, “click”, function)**

This adds listened to google Map

**infoWindow.open(map)**

It causes the window in google map to open

≪AJAX≫

request = new XMLHttpRequest();

request.open(“GET”, url);

request.onload = function(){

if(request.status == 200){

//do something

var data = request.responseText;

}

};

request.send(null);

**request.status == 200**

**request.readyState == 4**

**request.onload** is same as **request.onreadystatechange**

**JSON.stringify($array)**

Turns array into an object 🡪 JSON Format

**JSON.parse($object)**

Turns object into an array 🡪 JSON Format

≪Canvas≫

**var context = canvas.getContext("2d");**

Needs 2D context from the canvas to draw

**context.fillRect(intialPosX, initialPosY, width, height);**

The first two parameters are the x, y position of the rectangle

The last two parameters are the width and height of the rectangle

**context.fillStyle = “lightblue”**

It is a property fills out the color of the shapes.

**context.beginPath()**

Starting a new path

**context.moveTo(100, 50)**

This pin points the location where the line should start

**context.lineTo(250, 75)**

This method traces a path from the pencil’s current location to another point on the canvas.

Therefore, if the pencil’s initial point was {context.moveTo(100,50)} then it created a line between (100, 50) 🡪 (250, 75)

**Another context.lineTo(125, 30)** will create a line from (250, 75) 🡪 (125, 30)

Therefore, we have a line from (100, 50) 🡪 (250, 75) then another line from (250, 75) 🡪 (125, 30)

**context.closePath()**

This connects the starting point of the path to the last point in the current path

Therefore, (125, 30) 🡪 (250, 75)

**context.beginPath()**

Starts a new path

**context.arc(intialPosX, initialPosY, radius, start∠, end∠, true)**

The first two parameters are the initial position

The third parameter is the radius

The fourth and fifth parameter is the starting and ending angle in radians

The fifth parameter is the direction whether it is clockwise (false), counterclockwise (true)

**context.fill()**

It is used to fill the color in the path

**context.textAlign = “left”**

This property specifies where the anchor point for the text is. “start” is the default

Possible values are: start, end, left, right and center.

**context.fillText(text, intialPositionX, initialPositionY, maxWidth)**

**context.strokeText(text, intialPositionX, initialPositionY, maxWidth)**

The first parameter is the text

The second and the third parameters are positions of the text

The fourth parameter is the size of the text

**context.font = “2em Lucida Grande”**

**context.font = “italic bold 1.5em Times, serif”;**

**context.fillText(text, initialPositionX, initialPositionY)**

Adding fonts before adding the text.

**context.beginPath();**

**context.moveTo(100, 100);**

**context.lineTo(250, 100);**

**context.stroke();**

**context.textBaseline = “middle”;**

**context.fillText(“Alphabet”, 100, 100);**

The baseline property sets the alignment points in the font and determines the line our letters sit on. Possible values are top, hanging, alphabetic, ideographic and bottom

**context.save()**

Saving the context

**context.restore()**

Go back to initial state

**context.translate()**

Go back to initial state

**var image = new Image();**

**image.src = "icon.ico";**

**context.drawImage(image, IntialPosX, InitialPosY, width, height);**

The first parameter is the image

The second and third parameters are the initial position

The fourth and fifth parameter are the width and the height

≪Form≫

**var selectedIndex = shape.selectedIndex;**

**var shape = shape[selectedIndex].value;**

Getting the selected value from the form

**var selectedIndex = shape.selectedIndex;**

**var shape = shape[selectedIndex].value;**

Getting the selected value from the form

**var selection = document.getElementById(“mySelect”)**

**var option = document.createElement(“options”);**

**option.text = text;**

**option.value = value;**

**selection.options.add(option)**

Add Select options dynamically using javascript

≪Video≫

**buffer.drawImage(video, 0, 0, bufferCanvas.width, bufferCanvas.height);**

It takes an image and draws that image onto the canvas, at an x, y position for a given width and height.

The time we are getting an image from the video. By specifying the video as the source, drawImage gets one frame of the video as image data

**var frame = buffer.getImageData(0, 0, bufferCanvas.width, bufferCanvas.height);**

Then we grab the image data from the canvas context and store it in a variable, frame, so we can process it.

**video.addEventListener(“error”, errorHandler, false)**

It is used to handle the video errors

≪Local Storage≫

**localStorage.setItem(“key”, “value”);**

The Web Storage API is available to us through the localStorage object. It saves the data in key and value style

**localStorage.getItem(“key”);**

**localStorage["key"]**

To get the item

**parseInt(string)**

It converts string to INT

**localStorage.length**

It tells us how many items are there in the storage

**localStorage.key(numeric)**

It also saves in the numeric order

**localStorage.removeItem(key)**

Removes the local storage key

# SQL Concepts

A **database** is a container that holds tables and other SQL structures related to those tables.

A **column** is a piece of data stored by our table.

A **row** is a single set of columns that describe attributes of a single thing.

Columns and rows together make up a table

The word **field** is often used instead of column

The word **record** is often used instead of row

## CREATE DATABASE

**CREATE DATABASE table\_name**;

**USE DATABASE table\_name**;

Capitalization and underscores help our program in SQL (even though SQL doesn’t need them)

## CREATE TABLE

**CREATE TABLE table\_name (column\_name DATATYPE, … )**

## DATATYPES

CHAR(CHARACTER)

It’s a rigid and prefers its data to be a set length

INT(INTEGER)

The number should be whole and it can be negative too

BLOB

It takes large amount of text data

DATE

It keeps track of our dates and time

VARCHAR

It holds text data of up to 255 characters in length

DATETIME or TIMESTAMP

Keeps track of the dates

DEC(DECIMAL)

It gives numbers with decimal places

### DEC(3,2)

The first argument is the total number of digits of precision, and the second argument is the number of digits after the decimal point.

## DESCRIBE

**DESC table\_name**

Describes the structure of the table

## DROP TABLE

**DROP TABLE table\_name**

It deletes the table and any data in it

## INSERT DATA

**INSERT INTO table\_name(column name, …)**

**VALUES(‘value1’, …)**

**INSERT INTO table\_name**

**VALUES(‘value1’, …)**

### INSERTING DATA

Insert data with single quotes on it.

**Handle quotes with a backlash**

INSERT INTO table\_name VALUES(‘Grover**\’s** MILL’, ‘NJ’)

**Handle quotes with an extra single quote**

INSERT INTO table\_name VALUES(‘Grover**’’s** MILL’, ‘NJ’)

## INNER NULL

CREATE TABLE table\_name (column\_name DATATYPE **NOT NULL**, …)

## DEFAULT

CREATE TABLE table\_name (column\_name **DEFAULT 1.00**)

Using a **default** value fills the empty columns with a specified value

## SELECT

**SELECT \* from table\_name**

Select all the data from all the columns from table\_name

### WHERE

SELECT \* FROM table\_name **WHERE column\_name = ‘column\_data’**

Select all the data from the selected column where that column consists of selected data

The **VARCHAR, CHAR, BLOB, DATA** and **TIME** data types need single quotes. The numeric types **DEC** and **INT**, do not

### SPECIFIC COLUMN

**SELECT column\_name, column\_name, … FROM table\_name**

### BOOLEAN

#### AND

SELECT column\_name FROM table\_name WHERE column\_name = ‘column\_data’

**AND** column\_name = ‘column\_data’

#### OR

SELECT column\_name FROM table\_name WHERE column\_name = ‘column\_data’

**OR** column\_name = “column\_data’

#### NOT

SELECT column\_name FROM table\_name **WHERE column\_name NOT IN (‘column\_data\_1’, ‘column\_data\_2’)**

SELECT column\_name FROM table\_name **WHERE NOT column\_name BETWEEN number\_1 and number\_2**

SELECT column\_name FROM table\_name **WHERE NOT column\_name LIKE ‘A%’**

### COMPARISON OPERATOR

#### Greater than >

SELECT column\_name FROM table\_name column\_name = ‘column\_data’ AND

column\_name **>** column\_value

#### Less than <

SELECT column\_name FROM table\_name column\_name = ‘column\_data’ AND

column\_name **<** column\_value

#### Less than and Equals <=

SELECT column\_name FROM table\_name column\_name = ‘column\_data’ AND

column\_name **<=** column\_value

#### Greater than and Equals >=

SELECT column\_name FROM table\_name column\_name = ‘column\_data’ AND

column\_name **>=** column\_value

### NULL

SELECT column\_name FROM table\_name WHERE column\_name **IS NULL**

### LIKE

It is a wildcard operator

SELECT \* FROM table\_name WHERE column\_name **LIKE ‘%CA’**

The percent is a stand-in for any number of unknown characters

SELECT \* FROM table\_name WHERE column\_name **LIKE ‘\_CA’**

The underscore is a stand-in for just one unknown character

### BETWEEN

SELECT column\_table FROM table\_name WHERE column\_name **BETWEEN number\_value\_1 and number\_value\_2**

### IN

SELECT column\_table FROM table\_name WHERE **column\_name IN (‘data\_1’, ‘data\_2’)**

## DELETE

We can use WHERE clauses with DELETE statements the same way we use them with INSERT statements

**DELETE FROM table\_name WHERE table\_column = ‘table\_data’**

## UPDATE

**UPDATE table\_name SET column\_name = ‘column\_data’ WHERE column\_name = ‘column\_data’**

Update statements can replace DELETE/INSERT combinations. Update statements can be used on multiple records in our table. We can use them with basic math operators to manipulate our numeric values

UPDATE table\_name **SET column\_name = column value + 1**

WHERE column\_name = ‘column\_data’

OR column\_name = ‘column\_data’

UPDATE table\_name **SET column\_name = RIGHT(another\_column\_name, 2)**

## ATOMIC DATA

A little piece of information that can’t or shouldn’t be divided. Data should be broken into the smallest pieces of data that can’t or shouldn’t be divided.

**Rule 1:** A column with atomic data can’t have several values of the same type of data in that column

**Rule 2:** A table with atomic data can’t have multiple columns with the same type of data.

## NORMAL

Making our atomic is the first step in creating a NORMAL table.

**Benefit 1:** Normal tables won’t have duplicate data, which will reduce the size of our database

**Benefit 2:** With less data to search through our queries will be faster

Each row of data must contain atomic values

Each row of data must have a unique identifies, known as a PRIMARY KEY

## PRIMARY KEY

A primary key is a column in our table that makes each record unique.

**Rule 1:** A primary key can’t be NULL

**Rule 2:** A primary key must be given a value when the record is inserted

**Rule 3:** The primary must be compact

**Rule 4:** The primary key values can’t be changed

CREATE TABLE table\_name(

**column\_name\_1 DATATYPE NOT NULL AUTO\_INCREMENT,**

column\_name\_2 DATATYPE,

**PRIMARY KEY (column\_name\_1)**

)

Auto Increments just automatically fill the column with value that starts on row 1 with a value of 1 and goes up in increments of 1.

## SHOW

**SHOW CREATE TABLE table\_name**

## BACKTICK

It allows us to use whatever field name we wish when designing your table. Sometimes it makes a lot of sense to name a field `key`, `order`, or `values`... all of which require backticks when referring to them.

## ALTER

CHANGE: Changes both the name and data type of an existing column

MODIFY: Changes the data type or position of an existing column

ADD: Add a column to our table and it needs data type

DROP: Drop the column from the table.

### RENAME

ALTER TABLE **old\_table\_name RENAME TO new\_table\_name**

### CHANGE

ALTER TABLE table\_name **CHANGE COLUMN old\_column\_name new\_column\_name** INT NOT NULL AUTO\_INCREMENT, **ADD PRIMARY KEY** **(new\_column\_name)**

ALTER TABLE table\_name **CHANGE** **COLUMN old\_column\_name new\_column\_name** **NEW\_DATA\_TYPE**

### MODIFY

ALTER TABLE table\_name **MODIFY COLUMN column\_name DATATYPE**

### ADD

ALTER TABLE table\_name **ADD COLUMN column\_name DATATYPE AFTER specific\_column\_name**

**ALTER TABLE table\_name ADD COLUMN column\_name DATATYPE FIRST**

### DROP

ALTER TABLE table\_name **DROP COLUMN column\_name**

## STRING FUNCTION

Text values and values stored in CHAR or VARCHAR columns are known as strings. String functions allow us to select part of a text column.

We can use string functions in combination with SELECT, UPDATE, and DELETE

### RIGHT

SELECT **RIGHT(column\_name, count\_number\_from\_right)** FROM table\_name

### SUBSTRING

SELECT **SUBSTRING(COLUMN\_NAME, start\_position, length)** FROMtable\_name

### SUBSTR

UPDATE table\_name SET column\_name = **SUBSTR(column\_name\_1, LENGTH(column\_name\_2) + 2)**

### SUBSTRING INDEX

SELECT **SUBSTRING\_INDEX(column\_name, ‘delimiter’, which\_one)** FROM table\_name

**which\_one,** if it is “1”, it means it is looking for the first comma. If it were “2” it would keep going until finds a second comma and grab everything in front of that.

### UPPER

SELECT **UPPER(column\_name)** FROM table\_name

### LOWER

SELECT **LOWER(column\_name)** FROM table\_name

### LTRIM

SELECT **LTRIM(column\_name)** FROM table\_name

Removes any string with extra spaces removed from the left

### RTRIM

SELECT **RTRIM(column\_name)** FROM table\_name

Removes any string with extra spaces removed from the right

### REVERSE

SELECT **REVERSE(column\_name)** FROM table\_name

Reverses the order of letters in the string

### LENGTH

SELECT **LENGTH(column\_name)** FROM table\_name

Returns a count of hominy characters are in the string

## CONDITIONS

UPDATE table\_name

SET column\_name =

**CASE**

**WHEN column\_name\_1 = some\_value\_1**

**THEN new\_value\_1**

**WHEN column\_name\_2 = some\_value\_2**

**THEN new\_value\_2**

**ELSE new\_value\_3**

**END**

## ORDER BY

SELECT column\_name FROM table\_name WHERE column\_name = ‘column\_data’ **ORDER BY column\_name**

It allows us to alphabetically order any columns

### ORDER TWO COLUMNS

**SELECT** **column\_name\_1, column\_name\_2 FROM table\_name** **ORDER BY order\_column\_name\_1, order\_column\_name\_2**

column\_name\_1 will get ordered by order\_column\_name\_1

column\_name\_2 will get ordered by order\_column\_name\_2

### ORDER WITH MULTIPLE COLUMNS

**SELECT \* FROM table\_name ORDER BY order\_column\_name\_1, order\_column\_name\_2, order\_column\_name\_3**

First the columns are ordered by order\_column\_1 then order\_column\_2 and at the end order\_column\_3

We can sort by as many columns as we need

### SQL RULES OF ORDER

Non-alphabet characters show up before and after numbers.

Numbers show up before text characters.

NULL values show up before numbers and alphabet characters

Uppercase characters show up before and lowercase letters

A 1 will show up before A1

Order: **!= & ( \* + - ? @ ~**

### DESC

Keyword DESC after the column name in ORDER BY clause reverses the order of the result

SELECT \* FROM table\_name ORDER BY order\_column\_name\_1 **DESC**

### ASC

SELECT \* FROM table\_name ORDER BY order\_column\_name\_1 **ASC**

We can put ASC but it is not the necessary since the default order is ASC

## ARITHMETIC FUNCTION

### SUM

SELECT **SUM(column\_name)** FROM table\_name WHERE another\_column\_name = “column\_data”

### GROUP BY

SELECT **column\_name\_1,** **SUM(column\_name\_2)** FROM table\_name **GROUP BY column\_name\_1 ORDER BY SUM(column\_name\_2) DESC**

### AVERAGE

SELECT **column\_name\_1,** **AVG(column\_name\_2)** FROM table\_name **GROUP BY column\_name\_1**

### MIN AND MAX

SELECT **column\_name\_1,** **MAX(column\_name\_2)** FROM table\_name **GROUP BY column\_name\_1**

SELECT **column\_name\_1,** **MIN(column\_name\_2)** FROM table\_name **GROUP BY column\_name\_1**

### COUNT

SELECT **COUNT(column\_name)** FROM table\_name

It will return the number of rows in a column

### DISTINCT

SELECT **DISTINCT(column\_name)** FROM table\_name ORDER BY **another\_column\_name**

SELECT **COUNT(DISTINCT column\_name)** FROM table\_name

### LIMIT

SELECT **column\_name FROM table\_name LIMIT 2**

Lets only the first two data to show up

SELECT **column\_name FROM table\_name LIMIT 0,5**

Let us see the data from row 0 to row 5

## COMMENT

**/\*Select all the columns**

**of all the records**

**in the Customers table:\*/**

**--SELECT \* FROM Customers;**

## SCHEMA

A descript of the data (the columns and tables) in our database, along with any other related objects and the way they all connect is known as **schema**

Creating a diagram of our table lets us keep the design of the table separate from the data that inside of it.

## FOREIGN KEY

The **foreign key** is a column in a table that references the **primary key** of another table

We will be able to insert values into our foreign key that exist in the table the key came from, the parent table. This is called **referential integrity.**

Creating a foreign key as a constraint in our table give us definite advantages. We’ll get errors if we violate the rules, which will stop us accidentally doing anything to break the table.

Example:

**CREATE TABLE interest (**

--Adding the primary key command to the line where we set it up is quicker way to designate

**int\_id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

**interest VARCHAR(50) NOT NULL,**

**contact\_id INT NOT NULL,**

--The key came from (my\_contacts) 🡪 contact\_id and that it’s a foreign key

**CONSTRAINT my\_contacts\_contact\_id\_fk**

**FOREIGN KEY (contact\_id)**

**REFERENCES my\_contact (contact\_id)**

## RELATIONSHIP BETWEEN TABLES

|  |  |
| --- | --- |
|  | One to One |
|  | One to Many |
|  | Many to Many |

### ONE-TO-ONE

**TABLE B**

**TABLE A**

**ONLY ONE** of these rows -- Matches TO -- **ONLY ONE** of these rows

One-to-One: Exactly one row of a parent table is related to one row of child table

### ONE-TO-MANY

**TABLE B**

**TABLE A**

**ONLY ONE** of these rows -- Matches TO -- **MANY** of these rows

One-to-Many: A record in TABLE A can have many matching records in TABLE B, but a record in TABLE B can only match one record in TABLE A

### MANY-TO-MANY

**TABLE B**

**TABLE A**

**MANY** of these rows -- Matches TO -- **MANY** of these rows

### JUNCTION TABLE

Many-to-Many 🡪 A junction table holds a key from each table

## NORMAL FORM REVISITED

### ATOMIC DATA

A little piece of information that can’t or shouldn’t be divided. Data should be broken into the smallest pieces of data that can’t or shouldn’t be divided.

**Rule 1:** A column with atomic data can’t have several values of the same type of data in that column

**Rule 2:** A table with atomic data can’t have multiple columns with the same type of data.

### NORMAL

Making our atomic is the first step in creating a NORMAL table.

**Benefit 1:** Normal tables won’t have duplicate data, which will reduce the size of our database

**Benefit 2:** With less data to search through our queries will be faster

Each row of data must contain atomic values

Each row of data must have a unique identifies, known as a PRIMARY KEY

### FIRST NORMAL FORM

**1NF**

**Rule 1:** Columns contain only atomic values

**Rule 2:** No repeating groups of data

A key made of two or more columns is known as a **composite key**. A composite key is a primary key composed of multiple columns, creating a unique key. When a column’s data must change when another column’s data is modified

When a column’s data must change when another column’s data is modified, the first column is **functionally dependent** on the second.

**{Name + Power} 🡺 Composite Key**

|  |  |  |
| --- | --- | --- |
| **Name** | **Power** | **Weakness** |
| Superman | Fly | Dumb |
| Batman | Rope | Gadgets Dependent |
| Spiderman | Web | Monster |

**Shorthand Notations**

**T.x ->; T.y**

Technical term for this a shorthand notation

Relational Table is called T

Column y is functionally dependent on column x

**Partially functional dependency:** It means that a non-key column is dependent on some, but not all, of the columns in a composite primary key.

**{Name + Power} 🡺 Composite Key {Name is dependent 🡪 Initials}**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Power** | **Weakness** | **Initials** |
| Superman | Fly | Dumb | SM |
| Batman | Rope | Gadgets Dependent | BM |
| Spiderman | Web | Monster | SM |

**Transitive function dependency:** If changing any of the non-key columns might cause any of the other columns to change, we have a transitive dependency.

When any non-key column is related to any of the other non-key columns.

arch\_enemey\_id 🡺 arch\_enemy\_city

|  |  |  |
| --- | --- | --- |
| **Name** | **arch\_enemy\_id** | **arch\_enemy\_city** |
| Superman | 4 | La La City |
| Batman | 5 | Arkham City |
| Spiderman | 8 | Times City |

### SECOND NORMAL FORM

**2NF**

**Rule 1:** Be in 1NF

**Rule 2:** Have no partial functional dependencies

Your 1NF table is also 2NF, if all the columns in the table are part of the primary key or It has a single column primary key.

### THIRD NORMAL FORM

**3NF**

**Rule 1:** Be in 2NF

**Rule 2:** Have no transitive dependencies

If your table has an artificial primary key and no composite primary key, it’s in 2NF

### SUMMARY NORMAL FORM

1. A column with atomic data can’t have several values of the same type of data in that column
2. A table with atomic data can’t have multiple columns with the same type of data.
3. No repeating groups of data
4. Have no partial functional dependencies
5. Have no transitive dependencies

## MULTIPLE QUERIES

### CREATE TABLE, THEN INSERT WITH SELECT

CREATE TABLE profession (id INT(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY, profession varchar(20))

* Fill up the **profession column** of the **profession table** with the values from our **SELECT**

**INSERT INTO profession (profession)**

new\_**table new\_column**

**SELECT profession FROM my\_contacts**

existing\_column existing\_table

**GROUP BY profession**

**ORDER BY profession**

### CREATE TABLE WITH SELECT, THEN ALTER TO ADD PRIMARY KEY

* Create the **profession table** with **one column**, full of the values from the **SELECT**

**CREATE TABLE profession AS**

**SELECT profession FROM my\_contacts**

**GROUP BY profession**

**ORDER BY profession**

ALTER TABLE profession

ADD COLUMN id INT NOT NULL AUTO\_INCREMENT FIRST,

ADD PRIMARY KEY(id)

### CREATE, SELECT AND SELECT

* Create the profession table with both a primary key and a profession column, and fill the profession column with the values from the **SELECT**

**CREATE TABLE profession(**

**id INT(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

**profession varchar(20)**

**) AS**

**SELECT profession FROM my\_contacts**

**GROUP BY profession,**

**ORDER BY profession**

## ALIASES

Table aliases are also called **correlation names**

SELECT profession **AS mcprof**

FROM my\_contacts **AS mc**

**GROUP BY mc\_prof**

**ORDER BY mc\_prof**

Removing AS makes no difference but it means the same thing

SELECT profession **mcprof**

FROM my\_contacts **mc**

**GROUP BY msc\_prof**

**ORDER BY mc\_prof**

## CARTESIAN JOIN

The **cross-join** returns from one table crossed with every row from the second

**boy**

Davey

Bobby

**toy**

hula hoop

balsa glider

toy soldiers

harmonica

baseball cards

2\*5 = 10 combinations

|  |  |
| --- | --- |
| **toy** | **boy** |
| hula hoop | Davey |
| balsa glider | Bobby |
| toy soldiers | Davey |
| harmonica | Bobby |
| baseball cards | Davey |
| hula hoop | Bobby |
| balsa glider | Davey |
| toy soldiers | Bobby |
| harmonica | Davey |
| baseball cards | Bobby |

**SELECT t.toy, b.boy**

**FROM toy AS t**

**CROSS JOIN**

**boys as b**

### INNER JOIN

An **inner join** is a cross join with some result rows removed by a condition in the query.

An inner join combines the records from two tables using comparison operators in a condition

**SELECT column\_name FROM table\_1**

**INNER JOIN table\_2 ON some\_condition**

**Table Boys**

|  |  |  |
| --- | --- | --- |
| **boy\_id** | **boy** | **toy\_id** |
| 1 | Davey | 2 |
| 2 | Bobby | 1 |
| 3 | Beaver | 2 |

**Table Toys**

|  |  |
| --- | --- |
| **toy\_id** | **toy** |
| 1 | hula hoop |
| 2 | balsa glider |
| 3 | toy soldiers |

EQUIJOIN INNER JOIN

Test for equality

**SELECT boys.boy, toys.toy FROM boys**

**INNER JOIN toys**

**ON boys.toy\_id = toys.toy\_id**

**Result**

|  |  |
| --- | --- |
| **boy** | **Toy** |
| Beaver | balsa Glider |
| Bobby | hula hoop |
| Davey | balsa glider |

### NON EQUIJOIN INNER JOIN

test for inquality

**SELECT boys.boy, toys.toy FROM boys**

**INNER JOIN toys**

**ON boys.toy\_id <> toys.toy\_id**

**Result**

|  |  |
| --- | --- |
| **boy** | **Toy** |
| Beaver | hula hoop |
| Beaver | toy soldiers |
| Bobby | balsa glider |
| Bobby | toy soldiers |
| Davey | hula hoop |
| Davey | toy soldiers |

### NATURAL JOIN INNER JOIN

Identify matching column names, Column we’re joining by has the same name in both tables

**Result**

|  |  |
| --- | --- |
| **boy** | **Toy** |
| Beaver | balsa Glider |
| Bobby | hula hoop |
| Davey | balsa glider |

## SUBQUERY

A **subquery** is a query that is wrapped within another query. It’s also called INNER query.

OUTER QUERY + INNER QUERY = Query with a subquery

**OUTER QUERY**

**SELECT mc.firstname, mc.last\_name, cj.title**

**FROM current\_job AS cj**

**NATURAL JOIN my\_contacts as mc**

**WHERE cj.title IN (*SELECT title from job\_l istings*);**

***INNER QUERY***

SELECT mc.first\_name, mc.last\_ame, cj.salary

FROM my\_contact as mc

NATURAL JOIN current\_job as cj

WHERE **cj.salary = (SELECT MAX(cj.salary) FROM current\_job cj)**

If a **subquery is used as column expression** in a SELECT statement, it can only **return one** value from one column.

Therefore,

**SELECT mc.first\_name, mc.last\_name,**

**(SELECT state FROM zip\_code WHERE mc.zip\_code = zip\_code) AS state**

**FROM my\_contacts mc**

### NON-RELATED SUBQUERY

If the subquery stands alone and doesn’t reference anything from the outer query, it is a noncorrelated subquery.

**OUTER QUERY**

**SELECT mc.first\_name, mc.last\_name, cj.salary**

**FROM my\_contacts AS mc**

**NATURAL JOIN current\_job AS cj**

**WHERE cj.salary >**

**(SELECT cj.salary FROM my\_contacts AS mc**

**NATURAL JOIN current\_job AS cj**

**WHERE email = ‘andy@weathorma.com’)**

***INNER QUERY***

A non-correlated subquery uses **IN or NOT IN** to test if the values returned in the subquery are **members of a set (or not)**

SELECT mc.first\_name, mc.last\_name, cj.salary

FROM my\_contacts AS mc

NATURAL JOIN current\_job AS cj

**WHERE cj.title IN (SELECT title FROM job\_listings)**

SELECT mc.first\_name, mc.last\_name, cj.salary

FROM my\_contacts AS mc

NATURAL JOIN current\_job AS cj

**WHERE cj.title NOT IN (SELECT title FROM job\_listings)**

### CORRELATED SUBQUERY

The subquery depends on the outer query. It uses the same alias or correlation name that was created in outer query.

SELECT mc.first\_name, mc.last\_name FROM my\_contacts AS mc

**WHERE 3 = (SELECT COUNT(\*) FROM contact\_interest WHERE contact\_id = mc.contact\_id)**

### EXISTS

SELECT mc.first\_name firstname, mc.Last\_name lastname, mc.email email

FROM my\_contacts mc

**WHERE EXISTS**

**(SELECT \* FROM contact\_interest ci WHERE mc.contact\_id = ci.contact\_id)**

### NOT EXISTS

SELECT mc.first\_name firstname, mc.Last\_name lastname, mc.email email

FROM my\_contacts mc

**WHERE NOT EXISTS**

**(SELECT \* FROM current\_job cj WHERE mc.contact\_id = cj.contact\_id)**

## OUTER JOIN

### LEFT OUTER JOIN

The **left outer join** matches every row in the left row in the left table with a row from the right table.

**Table Toys**

|  |  |
| --- | --- |
| **toy\_id** | **toy** |
| 1 | hula hoop |
| 2 | balsa glider |
| 3 | toy soldiers |

**Table Boys**

|  |  |  |
| --- | --- | --- |
| **boy\_id** | **boy** | **toy\_id** |
| 1 | Davey | 2 |
| 2 | Bobby | 1 |
| 3 | Beaver | 2 |

**Result**

|  |  |
| --- | --- |
| **boy** | **toy** |
| Beaver | balsa Glider |
| Davey | balsa glider |
| Bobby | hula hoop |
| NULL | toy soldiers |

**SELECT b.boy, t.toy**

**FROM toys t**

**LEFT OUTER JOIN boys b**

**ON b.toy\_id = t.toy\_id**

So, the left outer join takes all the rows in the left table (the boys table) and matches them to rows in the RIGHT table (the toys table)

A **NULL** value in the results of a left outer join means that the right table has no values that correspond to the left table.

### RIGHT OUTER JOIN

The right outer join evaluates the right table against the left table

|  |  |
| --- | --- |
| **boy** | **Toy** |
| Beaver | balsa Glider |
| Bobby | hula hoop |
| Davey | balsa glider |

**SELECT b.boy, t.toy**

**FROM toys t**

**RIGHT OUTER JOIN boys b**

**ON b.toy\_id = t.toy\_id**

## SELF-REFERENCING FOREIGN KEY

A **self-referencing foreign key** is the primary key of a table used in that same table for another purpose.

## SELF JOIN

The **self-join** allows us to query a single table as though there were two tables with exactly the same information in them.

Boss Table

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **boss\_id** |
| 1 | Elsie | 3 |
| 2 | Snuggles | 1 |
| 3 | Mr. Hobo | 1 |
| 4 | Clarabelle | 4 |

**SELECT c1.name, c2.name AS boss**

**FROM boss\_table c1**

**INNER JOIN boss\_table c2**

**ON c1.boss\_id = c2.id**

**RESULT: Clarabelle**

## UNION

SELECT title FROM job\_current

**UNION**

SELECT title FROM job\_desired

**UNION**

SELECT title FROM job\_listings

Union can only take one ORDER BY at the end of the statement. This because UNION concatenates and groups the results from the multiple SELECT statements

Union Rules

**Rule 1:** The number of columns in each select statement must match.

**Rule 2:** It must also have the same expression ang aggregate function sin each select statement

**Rule 3:** Statements can be any order and it suppresses duplicate values from the results

**Rule 4:** The data type in the column need to either be the same or be convertible to each other

**Rule 5:** If we see duplicate data then we can use **UNION ALL**

### UNION ALL

SELECT title FROM job\_current

**UNION ALL**

SELECT title FROM job\_desired ORDER BY title

Removes any duplicate data

### CREATE TABLE UNION

**CREATE TABLE my\_union AS**

**SELECT title from job\_current UNION**

**SELECT title from job\_desired UNION**

**SELECT title from job\_listings**

## INTERSECT

CREATE title FROM job\_current

**INTERSECT**

SELECT title FROM job\_desired

Show the data that are common between these tables

## EXCEPT

SELECT title FROM job\_current

**EXCEPT**

SELECT title FROM job\_desired

Show all the data from job\_current except that are common with job\_desired

## CHECK

A **check** constraint restricts what values we can insert into a column. It uses the same conditionals as a WHERE clause

CREATE TABLE piggy\_bank(

id INT AUTO\_INCREMENT NOT NULL PRIMARY KEY,

**coin CHAR(1) CHECK (coin IN(‘P’,’N’,’D’,’Q’))**

)

### ALTER TABLE FOR CONSTRAINT

**ALTER TABLE my\_contacts**

**ADD CONSTRAINT CHECK gender IN (‘M’,’F’)**

## VIEW

**CREATE VIEW web\_designers AS**

SELECT mc.first\_name, mc.last\_name, mc.phone, mc.email

FROM my\_contacts mc

NATURAL JOIN job\_desired jd

WHERE jd.title = ‘Web Designer’

A view is basically a table that only exists when we use the view in a query. It’s considered a virtual table because it acts like table, and the same operations that can be performed on a table can be performed on a view

1. We can keep changes to our database structure from breaking applications that depend on our tables
2. View makes our life easier by simplifying our complex query into a simple command
3. We can create views that hide information that isn’t needed by the user

### CHECK OPTION

It checks each query we try to **INSERT** or **UPDATE** to see if it’s allowed according to the **WHERE** clause in our **view table**. In MySQL, we can imitate a **CHECK CONSTRAINT using a CHECK OPTION**

An **updatable view** includes all the **NOT NULL** columns from the tables it references. A non-updatable view is a view that doesn’t include all the not null columns.

**CHECK** constraints and view both help maintain control when we have multiple users

### DROP VIEW

**DROP VIEW**

Drop the view that has been created

## TRANSACTION

A **transaction** is a set of SQL statements that accomplish a single unit of work. During a transaction, all of the SQL statement should be completed without any interference or else none of them should be completed

### ACID

**ATOMICITY**

All of the pieces of transaction must be completed, or none of them will be completed.

Scenario: Power Outage

**CONSISTENCY**

A complete transaction leaves the database in a consistent state at the end of the transaction.

Scenario: Money transferred to savings and then translated into cash

**ISOLATION**

Every transaction has a consistent view of the database regardless of other transaction taking place at the same time.

Scenario: Two person of the same account holder going to the booth at once, only person should be able to retrieve the money, the other have to wait.

**DURABILITY**

After the transaction, the database needs to save the data correctly and protect it from power outages or other threats.

Scenario: Power Outage 🡪 Her records saved to a different location than the main database (backup)

Small Note

**[ALTER TABLE table\_name ADD INDEX (last\_name] quick way to index**

**START TRANSACTION**

It keeps track of all the SQL that follow until we enter **COMMIT or ROLLBACK**

**COMMIT**

If we have got all our statements in place and everything looks good, we **COMMIT** to make it permanent

**ROLLBACK**

If something isn’t quite right, **ROLLBACK** reverse everything to the way it was before we typed **START TRANSACTION**

To make the transaction work with MySQL we need to use the correct **storage engine**

CREATE TABLE(… …)**ENGINE=MyISAM DEFAULT CHARSET=latin1**, this cannot be used.

We need to make sure that our storage engine is either **BDB or InnoDB**, there are two choices that support transactions.

**ALTER TABLE** table\_name **TYPE = InnoDB**

**START TRANSACTION**

SELECT \* FROM piggy\_bank

UPDATE piggy\_bank set coin = ‘Q’ WHERE coin = ‘P’

SELECT \* FROM piggy\_bank

**ROLLBACK**

SELECT \* FROM piggy\_bank

**START TRANSACTION**

SELECT \* FROM piggy\_bank

UPDATE piggy\_bank set coin = ‘Q’ WHERE coin = ‘P’

SELECT \* FROM piggy\_bank

**COMMIT**

SELECT \* FROM piggy\_bank

## PASSWORD PROTECT

**SET PASSWORD FOR ‘root@localhost’ = PASSWORD (‘rootpass’)**

## CREATE USER

**CREATE USER jack IDENTIFIED BY ‘jackinthebox’**

jackinthebox, over here is the password

## GRANT

We can control exactly what user can do to tables and columns with the **GRANT** statement

Giving permission to the user

**GRANT SELECT ON table\_name TO jack**

## REVOKE

Remove the privilege from the table

**REVOKE SELECT ON table\_name FROM jack**

### GRANT OPTION

**REVOKE GRANT OPTION ON**

**DELETE ON table\_name**

**FROM jack, jill**

Jack and Jill can delete things from the chores table, but they can’t give anyone else the delete privilege to anyone else.

Admin🡪 gives **DELETE** privilege 🡪 Jack

Jack 🡪 gives **DELETE** privilege 🡪 Jill

Admin 🡪 removes **DELETE** privilege from Jack

Therefore, Jill will also lose the **DELETE** privilege

### CASCADE

REVOKE DELETE ON table\_name FROM Jack **CASCADE**

Admin🡪 gives **DELETE** privilege 🡪 Jack

Jack 🡪 gives **DELETE** privilege 🡪 Jill

Admin 🡪 removes **DELETE** privilege from Jack

Therefore, Jill will also lose the **DELETE** privilege

### RESTRICT

REVOKE DELETE ON table\_name FROM Jack **RESTRICT**

Admin🡪 gives **DELETE** privilege 🡪 Jack

Jack 🡪 gives **DELETE** privilege 🡪 Jill

Admin 🡪 removes **DELETE** privilege from Jack

Admin will receive an error

## CREATE ROLE

**CREATE ROLE data\_entry**

**GRANT SELECT, SELECT ON table\_name TO data\_entry**

**GRANT data\_entry TO table\_name**

GRANT data\_entry TO table\_name **WITH ADMIN OPTION**

REVOKE data\_entry FROM table\_name **CASCADE**

REVOKE data\_entry FROM table\_name **RESTRICT**

## DROP ROLE

**DROP ROLE data\_entry**

## COMBINING CREATE USER AND GRANT

**GRANT SELECT ON table\_name**

**TO user\_name IDENTIFIED by ‘password’**

## ALL

Greater than all finds any values larger than the biggest value in the set

SELECT name, rating FROM restaurant\_ratings WHERE **RATING > ALL**

(SELECT rating FROM restaurant\_ratings WHERE rating >3 and rating <9)

Less than all finds any values smaller than the smallest value in the set

SELECT name, rating FROM restaurant\_ratings WHERE **RATING** **< ALL**

(SELECT rating FROM restaurant\_ratings WHERE rating >3 and rating <9)

## ANY

Greater than any finds any values larger than the smallest value in the set

SELECT name, rating FROM restaurant\_ratings WHERE **RATING > ALL**

(SELECT rating FROM restaurant\_ratings WHERE rating >3 and rating <9)

Less than any finds any values smaller than the largest values in the set

SELECT name, rating FROM restaurant\_ratings WHERE **RATING** **< ALL**

(SELECT rating FROM restaurant\_ratings WHERE rating >3 and rating <9)

## SOME

It works same as any

## BOOLEAN

We can true or false

## INT

|  |
| --- |
| **DATA TYPES** |
| TINYINT |
| SMALLINT |
| MEDIUMINT |
| BIGINT |

## DATE AND TIME

|  |  |
| --- | --- |
| **DATA TYPES** | **FORMAT** |
| DATE | YYYY-MM-DD |
| DATETIME | YYYY-MM-DD HH:MM:SS |
| TIMESTAMP | YYYYMMDDHHMMSS |
| TIME | HH:MM:SS |

SELECT **DATE\_FORMAT(column\_name, ‘%M Y’)** FROM table\_name

## TEMPORARY TABLE

This table just for the session, it deletes out automatically

**CREATE TEMPORARY TABLE** table\_name(

column\_name INT,

column\_name VARCHAR(40)

)

Shortcut

**CREATE TEMPORARY TABLE table\_name AS SELECT \* FROM table\_name**

## CASTING

|  |
| --- |
| **DATA TYPE** |
| CHAR() |
| DATE |
| DATETIME |
| DECIMAL |
| SIGNED [INTEGER] |
| TIME |
| UNSIGNED [INTEGER] |

**SELECT CAST (‘2005-01-01’ AS DATE)**

**SELECT CAST (2 AS DECIMAL)**

SELECT **CAST(column\_name, TYPE)** FROM table\_name

We **can’t use** cast, for **decimal to integer** conversion

## WHO AM I

**SELECT CURRENT\_USER**

## MATH FUNCTION

|  |  |  |  |
| --- | --- | --- | --- |
| **ABS(x)** | Returns the absolute | **RADIANS(x)** | Degrees to radians |
| **ACOS(x)** | Returns the arccosine | **RAND(x)** | Random floating value |
| **ASIN(x)** | Returns the arcsine | **ROUND(x)** | 1.23 → 1, 1.56 → 2 |
| **ATAN(x,y)** | Returns arctangent | **ROUND(x,y)** | y represents decimal places |
| **CEIL(x)** | 2.34 🡪 3 | **SIGN(x)** | Returns 1 when x positive |
| **COS(x)** | Return the cosine (rad) | **SIN(x)** | Sinusoidal |
| **EXP(x)** | Returns the exponential | **SQRT(x)** | Square root |
| **FLOOR(x)** | 1.32 🡪 1 | **TAN(x)** | Tangent |
| **LN(x)** | Natural logarithm | **TRUNCATE(x,y)** | Cuts down the value |
| **LOG(x)** | Natural log with base 10 | **COT(x)** | Returns the cotangent |
| **LOG(x,y)** | Natural log with base y | **FORMAT(x,y)** | 3,451,000.50 |
| **MOD(x,y)** | Remainder of division |  |  |
| **PI()** | 3.142 |  |  |
| **POW(x,y)** | 3^2 = 9 |  |  |

# PHP Concepts

## PHP Novice Web Designing

### Introduction

With pure **HTML web pages**, the server simply serves **static HTML** that only display content. PHP is used to turn the web sites into interactive web application.



Fig 1: Web Server 🡪 Static HTML File 🡪 Client Web Browser

With **PHP** in the mix, the web server is able to **dynamically generate HTML** web pages





Fig 2: Web Server 🡪 PHP 🡪 MySQL 🡪 PHP 🡪 Client Web Server

**PHP**: **Personal Home Pages**, it was later changed **PHP Hypertext Processor**.

**PHP**: **Server-side programming language** – it runs on a web server.

#### Simple Form(Form.html)

<!doctype html>

<**html**>

<**head**>

<**meta** charset = "UTF-8"/>

<**title**> Simple Form </**title**>

</**head**>

<**body**>

<**p**> Please fill out the form </**p**>

<**img** src = "database.png" width = "100" height = "175"

alt = "database picture"/><**br**/>

<**form** method = "post" action = "khan.photon@gmail.com">

<**label** for = "firstname"> First Name: </**label**>

<**input** type = "text" id = "filename" name = "firstname"/><**br**/>

<**label** for = "lastname"> Last Name: </**label**>

<**input** type = "text" id = "lastname" name = "lastname"/><**br**/>

<**label** for = "email"> Email: </**label**>

<**input** type = "email" id = "email" name = "email"/><**br**/>

<**label** for = "gender">Gender: </**label**>

Male <**input** type = "radio" id ="gender" name = "gender" value = "male">

Female <**input** type = "radio" id = "gender" name = "gender"

value = female"><**br**/>

<**label** for = "description"> Description: </**label**>

<**textarea** name = "description"> Max word 500 words </**textarea**><**br**/>

<**input** type = "submit" value = "Submit" name = "submit"/>

</**form**>

</**body**>

</**html**>

Code 1 (Form.html): Simple Form HTML Code

**Tags:** <**form**></**form**>, <**input></input>,** <**label></label>**

**Attributes**: action, type, id, method, name, class, value

#### Breakdown(Form.html)

Basic HTML Format

<!doctype html>

<**html**>

<**head**>

<**meta** charset = "UTF-8"/>

<**title**> Simple Form </**title**>

</**head**>

<**body**>

</**body**>

</**html**>

|  |  |
| --- | --- |
| **Labels/Tags** | **Description** |
| <!doctype html> | A document type declaration |
| <**html**> | Tells the browser that this is an HTML document |
| <**head**></**head**> | Includes title for the document, scripts, styles, meta information |
| <**meta** charset="UTF-8"/> | Specifies what character set is our website written with |
| "UTF-8" | Universal Character Set + Transformation Format 8-bit |
| <**body**></**body**> | This is where all the HTML contents are written |
| <**p**></**p**> | Paragraph |
| <**img** src="database.png"/> | Include Image, Attribute src is the image src path |
| alt="database picture" | Alternatively, text will show up if no image found |
| width="100" height="175" | Fixing the width and height of the image |

<**form** method = "post" action = "khan.photon@gmail.com">

Form action attribute should be changed

<**form** method = "post" action = "report.php"></**form**>

Form tag causes the PHP script to run on the server.

|  |  |
| --- | --- |
| **Labels/Tags** | **Description** |
| <**input><**/**input>** | Input tag is the input field where user can enter data |
| type="text | Attribute type is the value received and saved from the user |
| id="filename" | Attribute id is the unique identifier; used for manipulating the element |
| name="firstname" | Attribute name is the value received and saved from the user |

<**label** for = "gender">Gender: </**label**>

Male <**input** type = "radio" id ="gender" name = "gender" value = "male">

Female <**input** type = "radio" id = "gender" name = "gender" value = "female">

Label tag allows user clicks on the text within the <**label>** element, it toggles the control.

For attribute should be equal to the id attribute of the related element to bind together

<**textarea** name = "description"> Max word 500 words </**textarea**><**br**/>

Textarea tag allows users to type multi-lines, description type situations

<**input** type = "submit" value = "Submit" name = "submit"/>

Type attribute allows user to make a submit button

Value attribute allows user to change the text in the submit button

<**br**/>

It allows the display go to next line. Single line break.

#### Confirmation(Report.php)

<?php

**if**(!empty($\_POST["submit"])):

$firstname = $\_POST["firstname"];

$lastname = $\_POST["lastname"];

$email = $\_POST["email"];

$gender = $\_POST["gender"];

$description = $\_POST["description"];

**echo** "First Name: ".$firstname."<br/>";

**echo** "Last Name: ".$lastname."<br/>";

**echo** "Email: ".$email."<br/>";

**echo** "Gender: ".$gender."<br/>";

**echo** "Description: ".$description."<br/>";

**endif**;

?>

Code 2 (Report.php): Receiving the user input information from form.html

#### Breakdown(Report.php)

|  |  |
| --- | --- |
| **Labels/Tags** | **Description** |
| <?php ?> | It indicates that the PHP code is written in this block |
| **If**(condition): | If the condition is true, do the following statement |
| $\_POST["submit"] | $\_POST receives the value from the name attribute |
| $firstname | $ creates the variable |
| **echo** | Outputting information beyond the confines of the <?php ?> |

Web browser know nothing about PHP and, therefore, have no ability to run PHP scripts.

Web servers with PHP support are equipped to run PHP scripts and turn them into HTML web pages that browsers can understand.

Every PHP must end with a semicolon “;”.

Name the PHP file with “.php”

#### PHP Information(info.php)

<?php

phpinfo();

?>

Code 3 (Info.php): Detailed information about the PHP installed in the PC

#### Variable

* PHP variable names must begin with a dollar sign, and cannot contain spaces
* The first character after the dollar be a letter or an underscore “\_”
* Characters after the first character after that can be a **letter, an underscore, or a number**
* Spaces and special characters are not allowed in any part of a variable name
* Use all lowercase for variable name
* Separate words in a multi-word variable name with underscores

#### Assigning value to the variable

$description = “I am the man.”;

Pieces of text(strings) must be enclosed by quotes, either single quotes or double quotes.

#### Special Variable

$\_POST is a special variable that is known as a superglobal. It is a collection of storage locations used to hold data from a web form. It is also an **array**. This array is filled the values the user entered into the form.

#### Concatenation

**echo** "Description: ".$description."<br/>";

The period allows us to stick multiple strings of text together as one. This process is known as concatenation.

<?php

$name = "Jack Stubbort";

$age = "17";

$place = "Netherland";

$msg = $name."is ".

$age."-years-old. "

."He works in ".$place.".";

**echo** $msg;

?>

Code 4 (Concatenation.php): Long line of PHP codes spanned across multiple lines

#### Escape Characters

* Escape characters in PHP starts with a backlash “\”
* Escape characters can be escaped in double-quoted only
* Single-quoted string only allow ‘\’ but not ‘\\’

<?php

$name = "Allen Smith";

$age = "21";

$occupation = "Engineer";

**echo** '$name is $age-years-old. \r\n He is an \"$occupation.\"';

**echo** "<br/><br/>";

**echo** nl2br("**$name** is **$age**-years-old. **\n** He is an **\"$occupation**.**\"\n\n**");

**echo** "**\\** **\"**";

?>

Code 5 (DoubleSingle.php): Escape Characters.

For newline break, we need to use nl2br() function to go new line.

### Mailing

#### Initial Setup

We can send mail from localhost with sendmail package, sendmail package is inbuild in XAMPP. So, if we are using XAMPP then you can easily send mail from localhost.

For example: We can configure C:\xampp\php\php.ini and c:\xampp\sendmail\sendmail.ini for gmail to send mail.

In C:\xampp\php\php.ini find extension=php\_openssl.dll and remove the semicolon from the beginning of that line to make SSL working for gmail for localhost.

In php.ini file find [mail function] and change

SMTP=smtp.gmail.com

smtp\_port=587

sendmail\_from = my-gmail-id@gmail.com

sendmail\_path = "\"C:\xampp\sendmail\sendmail.exe\" -t"

Now Open C:\xampp\sendmail\sendmail.ini. Replace all the existing code in sendmail.ini with following code

[sendmail]

smtp\_server=smtp.gmail.com

smtp\_port=587

error\_logfile=error.log

debug\_logfile=debug.log

auth\_username=my-gmail-id@gmail.com

auth\_password=my-gmail-password

force\_sender=my-gmail-id@gmail.com

Now you have done!! create php file with mail function and send mail from localhost.

PS: Don't forgot to replace my-gmail-id and my-gmail-password in above code. Also, don't forget to remove duplicate keys if you copied settings from above. For example comment following line if there is another sendmail\_path : sendmail\_path="C:\xampp\mailtodisk\mailtodisk.exe" in the php.ini file

Also remember to restart the server using the XAMMP control panel so the changes take effect.

For gmail please check [**https://support.google.com/accounts/answer/6010255**](https://support.google.com/accounts/answer/6010255%20) to allow access from less secure apps.

**Source:** [**https://stackoverflow.com/questions/15965376/how-to-configure-xampp-to-send-mail-from-localhost**](https://stackoverflow.com/questions/15965376/how-to-configure-xampp-to-send-mail-from-localhost)

#### Mailing(Mail.php)

<?php

**echo** '<!doctype html>';

**echo** ' <html>';

**echo** ' <head>';

**echo** ' <meta charset = "UTF-8"/>';

**echo** ' <title> Simple Form </title>';

**echo** ' </head>';

**echo** ' <body>';

**echo** ' <img src = "images/database.png" width = "100" height = "100"';

**echo** ' alt = "database picture"/><br/>';

**echo** ' <p> Please fill out the form </p>';

**echo** ' <form method = "post" action = "';

**echo** $\_SERVER['PHP\_SELF']."**\"**>";

**echo** ' <label for = "firstname"> First Name: </label>';

**echo** ' <input type = "text" id = "filename" name = "firstname"/><br/>';

**echo** ' <label for = "lastname"> Last Name: </label>';

**echo** ' <input type = "text" id = "lastname" name = "lastname"/><br/>';

**echo** ' <label for = "email"> Email: </label>';

**echo** ' <input type = "email" id = "email" name = "email"/><br/>';

**echo** ' <label for = "gender">Gender: </label>';

**echo** ' Male <input type = "radio" id ="gender" name = "gender" value = "male">';

**echo** ' Female <input type = "radio" id = "gender" name = "gender" value = "female"><br/>';

**echo** ' <label for = "description"> Description: </label>';

**echo** ' <textarea name = "description"> Max word 500 words </textarea><br/>';

**echo** ' <input type = "submit" value = "Submit" name = "submit"/>';

**echo** ' </form>';

**echo** ' </body>';

**echo** ' </html>';

?>

Code 6 (Mail.php): First Section.

Form.html file has been converted to PHP file by adding echo.

$\_SERVER['PHP\_SELF']

The form sends the data to the same page and not directing to another PHP file.

<?php

**if**(!**empty**($\_POST["submit"])):

$firstname = $\_POST["firstname"];

$lastname = $\_POST["lastname"];

$email = $\_POST["email"];

$gender = $\_POST["gender"];

$description = $\_POST["description"];

**echo** "First Name: ".$firstname."<br/>";

**echo** "Last Name: ".$lastname."<br/>";

**echo** "Email: ".$email."<br/>";

**echo** "Gender: ".$gender."<br/>";

**echo** "Description: ".$description."<br/>";

$to = "khan.photon@gmail.com";

$subject = "Extra Description";

$message = $description;

mail($to, $subject, $message, 'From:'.$email);

*//Allow less secure apps on*

**endif**;

?>

Code 7 (Mail.php): Second Section.

It is exactly like the Report.php file only the mail($to, $subject, $message, 'From:'.$email) function has been added.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| $to | Whom you are sending email to |
| $subject | Subject part of the email |
| $message | The actual message of the email |
| 'From:'.$email | Email Address of the sender |
| "From:".$email."\r\nCc:".$cc | Separate From and CC using (\r\n), Use double quotation |

### Connecting to the Database

SQL: Structured Query Language, it is the language used to communicate.

MySQL: Stores data inside of database tables. MySQL databases are organized into tables, which store information as rows and columns of related data.



Fig 3: Client Web Server 🡪 Request 🡪 Web server Response

#### Communicating with a Databaser Server

PhpMyAdmin: It is a graphical tool that allows us to create databases and tables through a web interface



Fig 4: PhpMyAdmin

First of all, install XAMPP for Windows or install LAMPP for Linux.

#### Installing XAMPP for Windows

Go to this link, [**https://www.apachefriends.org/download.html**](https://www.apachefriends.org/download.html), Download the latest version of XAMPP through the Mozilla browser.

After installation, open the XAMPP software and turn on



Fig 5: XAMPP Control Panel

Create a folder in **C:\xampp\htdocs\,** you can name folder anything, for the time being let us just create a folder named “practice”. Now we will have **C:\xampp\htdocs\practice,** we can create PHP or HTML files inside it and then deploy it on the browser. Let us create a file, **form.php**. Therefore, we will end up having **C:\xampp\htdocs\practice\form.php.**

Now go to your favorite browser, type [**http://localhost/practice/form.php**](http://localhost/practice/form.php)**,** code in PHP would display here. Therefore **C:\xampp\htdocs\** folder location gets replaced by [**http://localhost/**](http://localhost/) in your browser.

If we type, [**http://localhost/phpmyadmin/**](http://localhost/phpmyadmin/)**,** this will show our MySQL database.

#### Light Weight Editor (Sublime Text Editor)

From this link download, [**https://www.sublimetext.com/3**](https://www.sublimetext.com/3)**,** this is probably one of the best coding editor.

If you want to install packages in sublime text first you need to install **package control.** Press the **Ctrl+`** in sublime editor and copy paste the code given in [**https://packagecontrol.io/installation**](https://packagecontrol.io/installation)**.**



Fig 6: Sublime Text Editor (Package Control Install)

Once installed, press **Ctrl+Shift+P,** type in **Package Control: Install Package**

#### Installing XAMPP for Linux

Go to this link, <https://www.apachefriends.org/download.html>, Download the latest version of XAMPP.

Go to **/Home/Your-Username/Downloads.** Ex: **/Home/Photon/Downloads**

Open the command terminal there and type:

**sudo chmod 777 name-of-the-downloaded-file.**

Ex: **sudo chmod 777 ./xampp-linux-x64-7.2.2-0-installer.run**

This allows the administration permission to install the file

**sudo ./name-of-the-file**

Ex: **./xampp-linux-x64-7.2.2-0-installer.run**

This is how the installation of the file works

**sudo /opt/lamp/lampp start**

This starts the LAMPP on

#### Installing Sublime Editor for Linux

sudo apt-get-repository ppa:webupd8team/sublime-text-3

sudo apt-get update

sudo apt-get install sublime-text-installer

This is how the installation of the file works

### Creating a Database

Creating a database (form\_table). Tables are stored in database



Fig 7: Creating a Database (form\_table)

Click on New 🡪 Type in database name (form\_table) 🡪 click on create

#### Creating a Table

Tables serves as a way to divide up the data in a database into related groups



Fig 8: Creating a Table (Inside the form\_table Database)

#### Breakdown (SQL: form)

CREATE TABLE form (

id INT AUTO\_INCREMENT,

first\_name VARCHAR(30),

last\_name VARCHAR(30),

email VARCHAR(50),

gender VARCHAR(10),

description VARCHAR(100),

PRIMARY KEY(id)

)

Code 8: Create a form table

CREATE TABLE form

Name of the table is form

id INT AUTO\_INCREMENT

Number the rows using id, (integer and it will increment by itself)

first\_name VARCHAR(30)

First name will be stored here, its variable character length is 30

email VARCHAR(50)

Variable character length is 50

PRIMARY KEY(id)

This allows id to uniquely identify each record in the table

#### Using INSERT command

INSERT INTO form(first\_name, last\_name, email, gender, description)

VALUES('Photon', 'Khan', 'khan.photon@gmail.com', 'male', 'I am too fat')

Code 9: Insert data into the form table

INSERT INTO table\_name (column\_name, …. …. ….. ….)

VALUES (‘value 1’, ‘value 2’)

See the values are in single quotation. Values in the set of parentheses have to be in the same order as the database column names

#### Using SELECT command

SELECT columns FROM table\_name

Ex: SELECT first\_name, last\_name FROM form

SELECT \* FROM table\_name

Ex: SELECT \* FROM form

\*: Fetch all the data from all the columns in the table

Code 10: Select data from the table

### Automating SQL using PHP

Using PHP all the data entered through the form will be saved into the database.

<?php

**echo** '<!doctype html>';

**echo** ' <html>';

**echo** ' <head>';

**echo** ' <meta charset = "UTF-8"/>';

**echo** ' <title> Simple Form </title>';

**echo** ' </head>';

**echo** ' <body>';

**echo** ' <p> Please fill out the form </p>';

**echo** ' <form method = "post" action = "';

**echo** $\_SERVER['PHP\_SELF']."**\"**>";

**echo** ' <label for = "firstname"> First Name: </label>';

**echo** ' <input type = "text" id = "filename" name = "firstname"/><br/>';

**echo** ' <label for = "lastname"> Last Name: </label>';

**echo** ' <input type = "text" id = "lastname" name = "lastname"/><br/>';

**echo** ' <label for = "email"> Email: </label>';

**echo** ' <input type = "email" id = "email" name = "email"/><br/>';

**echo** ' <label for = "gender">Gender: </label>';

**echo** ' Male <input type = "radio" id ="gender" name = "gender" value = "male">';

**echo** ' Female <input type = "radio" id = "gender" name = "gender" value = "female"><br/>';

**echo** ' <label for = "description"> Description: </label>';

**echo** ' <textarea name = "description"> Max word 500 words </textarea><br/>';

**echo** ' <input type = "submit" value = "Submit" name = "submit"/>';

**echo** ' </form>';

**echo** ' </body>';

**echo** ' </html>';

?>

Code 11: Form.php (First Part)

<?php

**if**(!**empty**($\_POST["submit"])):

$firstname = $\_POST["firstname"];

$lastname = $\_POST["lastname"];

$email = $\_POST["email"];

$gender = $\_POST["gender"];

$description = $\_POST["description"];

**echo** "First Name: ".$firstname."<br/>";

**echo** "Last Name: ".$lastname."<br/>";

**echo** "Email: ".$email."<br/>";

**echo** "Gender: ".$gender."<br/>";

**echo** "Description: ".$description."<br/>";

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$database\_name = "form\_table";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $database\_name)

**or** **die**("Could not connect to the server");

$query = "INSERT INTO form(first\_name, last\_name, email, gender, description)

VALUES('**$firstname**','**$lastname**', '**$email**', '**$gender**', '**$description**')";

$result = mysqli\_query($connection, $query) **or** **die**("Error Querying database");

mysqli\_close($connection);

**endif**;

?>

Code 12: Form.php (Second Part)

#### Breakdown (Form.php (Second Part))

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Could not connect to the server");

$query = "INSERT INTO form(first\_name, last\_name, email, gender, description)

VALUES('**$firstname**','**$lastname**', '**$email**', '**$gender**', '**$description**')";

$result = mysqli\_query($connection, $query) **or** **die**("Error Querying database");

mysqli\_close($connection);

**endif**;

|  |  |
| --- | --- |
| **Terms** | **Explanation** |
| Mysqli | “i” over here stands for improved |
| mysqli\_connect | This allows us to connect to the database |
| $server\_name | It is “localhost” by default if we are running website “locally” |
| $user\_name | “root”, by default phpMyAdmin gives username as root |
| $password | “”, by default, there is no password |
| $db\_name | Database name |
| mysqli\_query() | It allows us to send the SQL command using PHP |
| mysqli\_close() | Connection should be closed or else the DB might get corrupted |
| **die**() | This function terminates a PHP script and gives the feedback of the error |

To be sure check your phpMyAdmin home page



Fig 9: Checking the server name, user name and the password

#### Database check in actual website

First go to **your-name-of-the-website/cpanel**

Ex: [**www.sphotonkhan/cpanel**](http://www.sphotonkhan/cpanel)

Before going to phpMyAdmin, go to MySQL® Database Wizard



Fig 10: MySQL Database Wizard (1st Step)



Fig 11: MySQL Database Wizard (2st Step)

|  |  |
| --- | --- |
| $server\_name | bh-40 |
| $user\_name | sphotonkhan |
| $password | photon |
| $db\_name | sphotonk\_test |

To get the server information go to Server Information

Fig 12: General Information 🡪 Server Information



Fig 13: Server Information

#### Selecting all the data from the table

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "form\_table";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Could not connect to the database");

$query = "SELECT \* FROM form";

$result = mysqli\_query($connection, $query)

**or** **die**("Query not working");

**while**($row = mysqli\_fetch\_array($result)){

$id = $row['id'];

$first\_name = $row['first\_name'];

$last\_name = $row['last\_name'];

$email = $row['email'];

$gender = $row['gender'];

$description = $row['description'];

**echo** nl2br("Name: **$first\_name** **$last\_name\n**");

**echo** nl2br("Email: **$email\n**");

**echo** nl2br("Gender: **$gender\n**");

**echo** nl2br("Description: **$gender\n\n**");

}

mysqli\_close($connection);

?>

Code 13: Select.php

### Web Application

A web application is a dynamic web site that designed to fulfill a particular purpose for it users.

A database is a container storing data in a very structured way. Table stores data in a grid-like pattern of columns and rows. Columns consists of specific type of data. Rows are collection of columns where a single row consists one of each column. Table rows are horizontal, and table columns are vertical.

**Application Purpose**: Customer wants to add into the newsletter program for a particular shop, i.e. get emails about the discounts and so on. Customer also have the authority to get removed from the newsletter program.

Add Email PHP Script PHP Engine Server Database

Send Email PHP Script PHP Engine Server Client



Database

Fig 14: Web Application

CREATE DATABASE is the SQL command used to create a new database.

CREATE DATABASE database\_name

Ex: CREATE DATABASE store

Text data takes more room to store than integer data. To create a table, we need to know what type of data is stored in each table column.

**Column Types**

CHAR: Character, Highly efficient for the texts with fixed length

INT: Integer, Whole Number, Negative numbers can also be stored

TINYINT: Small Integer

BLOB: Large gobs of binary data

DEC: Decimal

DATETIME: She keeps track of the date

TIMESTAMP: She keeps track of the date

DATE: Keep track of dates

VARCHAR: Variable Character

TEXT: Store huge amount of text

CREATE TABLE SQL is used to create a new table in a database.

CREATE TABLE table\_name (column name column type, …)

It is not possible to recreate the same table using CREATE TABLE

USE command selects a database as the default database for subsequent SQL statements

Ex: USE store

DESCRIBE reveals the structure of tables

Ex: DESCRIBE store

DROP deletes a table and all its data from the database

Ex: DROP TABLE store

#### Add Email Script

First Create Table

CREATE TABLE email\_list(

id INT AUTO\_INCREMENT,

first\_name VARCHAR(250),

last\_name VARCHaR(250),

email VARCHAR(60),

PRIMARY KEY(id)

)

<!doctype html>

<**html**>

<**head**>

<**title**> Add Email </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**p**> Enter Your Details </**p**>

<**form** method = "post" action = "addemail.php">

<**label** for = "first\_name"> First Name: </**label**>

<**input** type = "text" name = "first\_name" id = "first\_name"><**br**/>

<**label** for = "last\_name"> Last Name: </**label**>

<**input** type = "text" name = "last\_name" id = "last\_name"><**br**/>

<**label** for = "email"> Email: </**label**>

<**input** type = "email" name = "email" id = "email"><**br**/>

<**input** type = "submit" name = "submit" value = "Submit"/>

</**form**>

</**body**>

</**html**>

Code 14: addemail.html

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

**if**(!**empty**($\_POST['submit'])):

$connection = mysqli\_connect($server\_name,

$user\_name,

$password, $db\_name)

**or** **die**("Could not connect to the server");

$first\_name = $\_POST['first\_name'];

$last\_name = $\_POST['last\_name'];

$email = $\_POST['email'];

$sql = "INSERT INTO email\_list(first\_name,

last\_name,

email)

VALUES('**$first\_name**','**$last\_name**','**$email**')";

$result = mysqli\_query($connection, $sql)

**or** **die**("Query Denied");

**echo** nl2br("<h1>Confirmation Message</h1> **\n**");

**echo** nl2br("<p>Name: **$first\_name** **$last\_name**</p>");

**echo** nl2br("<p>Email: **$email**</p> **\n**");

mysqli\_close($connection);

**endif**;

?>

Code 15: addemail.php

<!doctype html>

<**htmL**>

<**head**>

<**title**> Send Email </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**form** method = "post" action = "sendemail.php">

<**label** for = "subject">Subject</**label**><**br**/>

<**input** type = "text" name = "subject" id = "subject"/><**br**/>

<**label** for = "body\_of\_email"> Body of Email </**label**><**br**/>

<**textarea** name = "body\_of\_email" id = "body\_of\_email"

rows = "8" cols = "60"></**textarea**><**br**/>

<**input** type = "submit" id = "submit" name = "submit"

value = "Submit"/>

</**body**>

</**htmL**>

Code 16: sendemail.html

<?php

**if**(!empty($\_POST['submit'])):

$from = "khan.photon@gmail.com";

$subject = $\_POST['subject'];

$body\_of\_email = $\_POST['body\_of\_email'];

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could connect to the server");

$query = "SELECT \* FROM email\_list";

$result = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

$to = $row['email'];

mail($to, $subject, $body\_of\_email, "From: **$from**");

}

mysqli\_close($connection);

**endif**;

?>

Code 17: sendemail.php

mysqli\_fetch\_array($result)

This function stores a row of data in an array.

A while loop repeats code while a condition is met. $row is interpreted as true since it isn’t set to 0 or false. When there is no more data available, mysqli\_fetch\_array($result) return false. Anything other than 0 or False is always interpreted as true.

#### Removing data using DELETE and WHERE

A where clause narrows down a query to focus on specific rows of data. A where clause in a DELETE statement let us pinpoint the row we want to remove.

DELETE FROM email\_list

WHERE email = “khan.photon@gmail.com”

email = Name of the table column

[khan.photon@gmail.com](mailto:khan.photon@gmail.com) = Check whether this exists in the database.

<!doctype html>

<**html**>

<**head**>

<**title**> Remove Email </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**p**> Enter Email Address to Remove </**p**><**br**/>

<**form** method = "post" action = "removeemail.php">

<**label** for = "email">Email</**label**><**br**/>

<**input** type = "email" name = "email" id = "email"/><**br**/>

<**input** type = "submit" value = "Submit" name = "submit"/>

</**form**>

</**body**>

<**html**>

Code 18: removeemail.html

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

**if**(!**empty**($\_POST)):

$email = $\_POST['email'];

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could not connect to the server");

$query = "DELETE FROM email\_list where email = '**$email**'";

mysqli\_query($connection, $query) **or**

**die**("Query Denied");

**echo** nl2br("**$email** has been successfully removed**\n**");

mysqli\_close($connection);

**endif**;

?>

Code 19: removeemail.php

#### Server-Side Validation

It assures the data we get is the data we expect.

isset(): It tests to see if a variable exists, which means that it’s been assigned a value.

empty(): Determines where the variable contains empty value

PHP defines as 0, an empty string (‘’ or “”), or the values false or NULL

PHP logic operators make it possible to structure more elegant if statements.

Logical AND is coded as && while logical OR is coded as ||

<!doctype html>

<**htmL**>

<**head**>

<**title**> Send Email </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**form** method = "post" action = "sendvalidatedemail.php">

<**label** for = "subject">Subject</**label**><**br**/>

<**input** type = "text" name = "subject" id = "subject"/><**br**/>

<**label** for = "body\_of\_email"> Body of Email </**label**><**br**/>

<**textarea** name = "body\_of\_email" id = "body\_of\_email"

rows = "8" cols = "60"></**textarea**><**br**/>

<**input** type = "submit" id = "submit" name = "submit"

value = "Submit"/>

</**body**>

</**htmL**>

Code 20: sendvalidatedemail.html

<?php

**if**(**empty**($\_POST['subject']) && !**empty**($\_POST['body\_of\_email'])):

**echo** "Your subject is empty!";

**elseif**(**empty**($\_POST['body\_of\_email']) && !**empty**($\_POST['subject'])):

**echo** "Your body of email is empty!";

**elseif**(**empty**($\_POST['subject']) && **empty**($\_POST['body\_of\_email'])):

**echo** "Your subject and body of email are empty!";

**elseif**(!empty($\_POST['submit']) && !**empty**($\_POST['subject']) && !**empty**($\_POST['body\_of\_email'])):

$from = "khan.photon@gmail.com";

$subject = $\_POST['subject'];

$body\_of\_email = $\_POST['body\_of\_email'];

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could connect to the server");

$query = "SELECT \* FROM email\_list";

$result = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

$to = $row['email'];

mail($to, $subject, $body\_of\_email, "From: **$from**");

}

mysqli\_close($connection);

**else**:

**endif**;

?>

Code 21: sendvalidatedemail.php

#### Self-Referencing and Sticky Forms

An HTML form that is part of the PHP script that processes it is known as **self-referencing.**

**Sticky forms** remember the data the user has already correctly entered.

$\_SERVER[‘PHP\_SELF’] stores away the name of the current script

<?php

$error = [];

$error['sub'] = "";

$error['body'] = "";

**if**(**empty**($\_POST['subject']) && !**empty**($\_POST['body\_of\_email'])):

$error['sub'] = "Your subject is empty!";

**endif**;

**if**(**empty**($\_POST['body\_of\_email']) && !**empty**($\_POST['subject'])):

$error['body'] = "Your body of email is empty!";

**endif**;

**if**(**empty**($\_POST['subject']) && **empty**($\_POST['body\_of\_email'])):

$error['sub'] = "Your subject is empty!";

$error['body'] = "Your body of email is empty!";

**endif**;

**if**(!**empty**($\_POST['submit']) && !**empty**($\_POST['subject']) && !**empty**($\_POST['body\_of\_email'])):

$from = "khan.photon@gmail.com";

$subject = $\_POST['subject'];

$body\_of\_email = $\_POST['body\_of\_email'];

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could connect to the server");

$query = "SELECT \* FROM email\_list";

$result = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

$to = $row['email'];

mail($to, $subject, $body\_of\_email, "From: **$from**");

**echo** "Your email has been successfully sent";

}

mysqli\_close($connection);

**endif**;

?>

<!doctype html>

<htmL>

<head>

<title> Send Email </title>

<meta charset = "UTF-8"/>

</head>

<body>

<form method = "post" action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<label for = "subject">Subject</label><br/>

<input type = "text" name = "subject" id = "subject"

value = "<?php **if**(!**empty**($\_POST['subject'])) **echo** $\_POST['subject']; ?>"/>

<?php **if**($error['sub'] && !**empty**($\_POST['submit'])) **echo** $error['sub'];?><br/>

<label for = "body\_of\_email"> Body of Email </label><br/>

<textarea name = "body\_of\_email" id = "body\_of\_email"

rows = "8" cols = "60"><?php **if**(!**empty**($\_POST['body\_of\_email'])) **echo** $\_POST['body\_of\_email'];?></textarea>

<?php **if**($error['body'] && !**empty**($\_POST['submit'])) **echo** $error['body'];?><br/>

<input type = "submit" id = "submit" name = "submit"

value = "Submit"/>

</body>

</htmL>

Code 22: sendemailupdated.php

#### Avoid duplicate entry in Database

Table rows should be **uniquely identifiable.** Therefore, we will create a unique integer column, also called a **primary key**

To alter the table that already exists

ALTER TABLE table\_name ADD column\_name, column\_type

Location: first in the table

ALTER TABLE email\_list ADD id INT NOT NULL AUTO\_INCREMENT FIRST, ADD PRIMARY KEY(id)

Column that we want to add Increase the value of id by 1 automatically id is the primary key

A primary key is a column in our table that makes each row unique

<!doctype html>

<html>

<head>

<title> Remove Email </title>

<meta charset = "UTF-8"/>

</head>

<body>

<p> Delete Email Addresses </p>

<?php

$showform = **TRUE**;

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name,

$user\_name,

$password,

$db\_name)

**or** **die**("Could not connect to the server");

**if**(isset($\_POST['submit'])):

$email = $\_POST['email'];

$showform = **FALSE**;

$query = "SELECT \* FROM email\_list WHERE email = '**$email**'";

$result = mysqli\_query($connection, $query) **or**

**die**("Select Query Denied");

**echo** "<form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**while**($row = mysqli\_fetch\_array($result)){

$id = $row['id'];

$name = $row['first\_name']." ".$row['last\_name'];

$email = $row['email'];

**echo** "<input type = 'checkbox' value = **\"$id\**" name = \"todelete[] **\"**/>";

**echo** "**$name** **$email**<br/>";

}

**echo** "<br/><input type = 'submit' value = 'Remove' name = 'remove'/>";

**echo** "</form>";

**endif**;

**if**(isset($\_POST['remove'])):

$showform = **FALSE**;

**foreach**($\_POST['todelete'] **as** $delete\_id){

$query = "DELETE FROM email\_list where id = '**$delete\_id**'";

mysqli\_query($connection, $query) **or** **die** ("Remove Query Denied");

}

**echo** "Customers Removed Successfully";

**endif**;

?>

<?php

**if**($showform):

?>

<form method = "post" action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<label for = "email">Email</label><br/>

<input type = "email" name = "email" id = "email"/><br/>

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

</form>

</body>

<html>

<?php

**endif**;

?>

Code 23: removeemailupdated.php

#### Inventory (ALTER TABLE)

Making a web application related to a shop

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Date Added** | **Product** | **Price** | **Image** |
| 1 | 2007-04-22 14:37:34 | Lip stick | 25.00 | --- |
| 2 | 2009-01-12 21:27:54 | Chocolate | 5.23 | --- |
| 3 | 2011-12-23 09:06:35 | Shampoo | 40.15 | --- |
| 4 | 2013-08-04 09:12:53 | Shoe | 150.14 | --- |
| 5 | 2012-02-21 14:09:50 | Soap | 9.19 | --- |

The Alter statement is used to change the structure of a database

ALTER TABLE inventory ADD COLUMN age TINYINT

Add a column to a table

ALTER TABLE inventory CHANGE COLUMN image screenshot VARCHAR(255)

Change the name and data type of the column

ALTER TABLE inventory DROP COLUMN age

Drops the column

ALTER TABLE inventory MODIFY COLUMN date DATETIME AFTER age

Only First & AFTER can be used to alter the position

Changes the data type or position of a column within a table

#### Uploading a File

<form enctype = "multipart/form=data" method = "post"

action = "<?php echo **$\_SERVER['PHP\_SELF']**;?>">

enctype attribute tells the form to use a special type of encoding required for the upload – it affects how the POST data is bundled and sent when the form is submitted

<input type = "hidden" name = "MAX\_FILE\_SIZE" value = "32768"/>

Establishes the maximum file size for file uploads in this case, 32 KB (32,768)

INSERT INTO inventory VALUES(NOW(), ‘$name’, ‘$price’, ‘$image’)

NOW() function is used to insert the current date/time.

Image filenames are stored in the database as part of an INSERT statement

$\_FILES

It is a built-in super global variable provides access to information about uploaded files.

$\_FILES[‘image’][‘name’]

The name of the uploaded file

$\_FILES[‘image’][‘type’]

MIME type of the uploaded file

$\_FILES[‘image’][‘tmp\_name’]

Temporary storage location of the file on the server

$\_FILES[‘error’][‘error’]

The error code for the file upload, 0 indicates a success, other values indicate failure

move\_uploaded\_file($\_FILES[‘image’][‘tmp\_name’], $target)

Placing an image on a web page only requires reference to the image file.

The file that is being uploaded actually stores in a temporary folder on the server. Temporary folder after a certain time deletes the file. Therefore. to save the file, we need to use move\_uploaded\_file() function to move the file from temporary to the current directory.

UPLOAD\_PATH = “images/”

It is a constant, it values stays the same

Define a constant using define(‘UPLOAD\_PATH’, ‘/images’)

$target = UPLOAD\_PATH.$FILES[‘image’][‘name]

Therefore, the target location is “images/name-of-the-file.png”

<!doctype html>

<html>

<head>

<title> Inventory Index </title>

<meta charset = "UTF-8"/>

</head>

<body>

<h1> Inventory **List** </h1>

<p><a href = "addinventory.php">Add Item</a></p>

<p> **List** of items are given below</p>

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

define("UPLOAD\_PATH","images/");

$connection = mysqli\_connect($server\_name, $user\_name,

$password, $db\_name)

**or** **die**("Could not conenct to the database");

$query = "SELECT \* FROM inventory";

$result = mysqli\_query($connection, $query);

**echo** "<table>";

**echo** "<tr>";

**echo** "<th>Date</th>";

**echo** "<th>Name</th>";

**echo** "<th>Price</th>";

**echo** "<th>Image</th>";

**echo** "</tr>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** "<td>".$row['date']."</td>";

**echo** "<td>".$row['name']."</td>";

**echo** "<td>".$row['price']."</td>";

**echo** "<td><img src ='".UPLOAD\_PATH.$row['image']."' alt = 'Inventory Image'

height = '20px' width = '20px'/></td>";

**echo** "</tr>";

}

**echo** "</table>";

?>

</body>

</html>

Code 24: inventoryindex.php

<?php

$show\_form = **TRUE**;

define("UPLOAD\_PATH","images/");

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Connection Denied");

**if**(isset($\_POST['submit'])):

$name = $\_POST['item\_name'];

$price = $\_POST['price'];

$image\_name = $\_FILES['image']['name'];

**if**(!**empty**($name) && !**empty**($price) && !**empty**($image\_name)):

$show\_form = **FALSE**;

$target = UPLOAD\_PATH.$image\_name;

*// $date = date("Y-m-d h:i:s");*

$query = "INSERT INTO inventory VALUES(0, NOW(), '**$name**', '**$price**', '**$image\_name**')";

mysqli\_query($connection, $query) **or** **die**("Query Denied");

move\_uploaded\_file($\_FILES['image']['tmp\_name'], $target);

**echo** "Inventory Added Successfully";

**endif**;

**endif**;

?>

<?php

**if**($show\_form):

?>

<!doctype html>

<html>

<head>

<title> Add Inventory </title>

<meta charset = "UTF-8"/>

</head>

<body>

<p> Add Inventory </p>

<form enctype = "multipart/form-data" method = "post"

action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<!-- <input type = "hidden" name = "MAX\_FILE\_SIZE" value = "32768"/> -->

<label for = "item\_name"> Item Name: </label><br/>

<input type = "text" name = "item\_name" id = "item\_name"/><br/>

<label for = "price"> Price: </label><br/>

<input type = "number" name = "price" id = "price" step = "0.01"/><br/>

<input type = "file" name = "image" id = "image"/><br/><br/>

<input type = "submit" value = "Add Item" name = "submit"/>

</body>

</html>

<?php

**endif**;

?>

Code 25: addinventory.php

#### Check File or not

<?php

$file\_path = "images/chips.png";

**if**(is\_file($file\_path) && filesize($file\_path)):

**echo** "It is a file";

**endif**;

?>

Code 26: checkfile.php

Databases are great for storing text data, but it’s normally better for them to reference binary data in external files.

#### Sharing Data

Shared script data needs to be accessible throughout an application without code duplication. Include files allows us to share code across multiple scripts

require\_once, the name “include file” comes from a PHP statement called include that is very similar to require\_once. The difference is that require\_once results in an error if the included file cannot be found. “include” won’t reveal an error if an include file is missing. There are in total four of them: include, include\_once, require, require\_once.

Therefore, **require** and **require\_once,** will shot **fatal error** if the file is missing

Therefore, **include** and **include\_once**, will show warning and continue load the rest of the page.

Therefore, difference between **include** and **include\_once**, if the code from a file has been already included then it will not be included again if we use **include\_once()**. Means **include\_once()** include the file only once at a time. Same for **require** and **require\_once.**

require\_once statement inserts shared script code into other scripts.

Let’s reorganize the previous files

<?php

define("UPLOAD\_PATH","images/");

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "store";

?>

Code 27: databaseinfo.php

<!doctype html>

<html>

<head>

<title> Inventory Index </title>

<meta charset = "UTF-8"/>

</head>

<body>

<h1> Inventory **List** </h1>

<p><a href = "addinventory.php">Add Item</a></p>

<p> **List** of items are given below</p>

<?php

**require\_once**("common/databaseinfo.php");

$connection = mysqli\_connect($server\_name, $user\_name,

$password, $db\_name)

**or** **die**("Could not conenct to the database");

$query = "SELECT \* FROM inventory";

$result = mysqli\_query($connection, $query);

**echo** "<table>";

**echo** "<tr>";

**echo** "<th>Date</th>";

**echo** "<th>Name</th>";

**echo** "<th>Price</th>";

**echo** "<th>Image</th>";

**echo** "</tr>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** "<td>".$row['date']."</td>";

**echo** "<td>".$row['name']."</td>";

**echo** "<td>".$row['price']."</td>";

**echo** "<td><img src ='".UPLOAD\_PATH.$row['image']."' alt = 'Inventory Image'

height = '20px' width = '20px'/></td>";

**echo** "</tr>";

}

**echo** "</table>";

?>

</body>

</html>

Code 28: updatedinventoryindex.php

<?php

**require\_once**("common/databaseinfo.php");

$show\_form = **TRUE**;

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Connection Denied");

**if**(isset($\_POST['submit'])):

$name = $\_POST['item\_name'];

$price = $\_POST['price'];

$image\_name = $\_FILES['image']['name'];

**if**(!**empty**($name) && !**empty**($price) && !**empty**($image\_name)):

$show\_form = **FALSE**;

$target = UPLOAD\_PATH.$image\_name;

*// $date = date("Y-m-d h:i:s");*

$query = "INSERT INTO inventory VALUES(0, NOW(), '**$name**', '**$price**', '**$image\_name**')";

mysqli\_query($connection, $query) **or** **die**("Query Denied");

move\_uploaded\_file($\_FILES['image']['tmp\_name'], $target);

**echo** "Inventory Added Successfully";

**endif**;

**endif**;

?>

<?php

**if**($show\_form):

?>

<!doctype html>

<html>

<head>

<title> Add Inventory </title>

<meta charset = "UTF-8"/>

</head>

<body>

<p> Add Inventory </p>

<form enctype = "multipart/form-data" method = "post"

action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<!-- <input type = "hidden" name = "MAX\_FILE\_SIZE" value = "32768"/> -->

<label for = "item\_name"> Item Name: </label><br/>

<input type = "text" name = "item\_name" id = "item\_name"/><br/>

<label for = "price"> Price: </label><br/>

<input type = "number" name = "price" id = "price" step = "0.01"/><br/>

<input type = "file" name = "image" id = "image"/><br/><br/>

<input type = "submit" value = "Add Item" name = "submit"/>

</body>

</html>

<?php

**endif**;

?>

Code 29: updatedinventoryadd.php

#### Order the table data

SELECT \* FROM inventory ORDER BY ASC, price DESC;

<!doctype html>

<htmL>

<head>

<title> Order Inventory </title>

<meta charset = "UTF-8"/>

</head>

<body>

<?php

**require\_once**("common/databaseinfo.php");

$connection = mysqli\_connect($server\_name, $user\_name,

$password, $db\_name)

**or** **die**("Could not conenct to the database");

$query = "SELECT \* FROM inventory ORDER BY price DESC";

$result = mysqli\_query($connection, $query);

**echo** "<table>";

**echo** "<tr>";

**echo** "<th>Date</th>";

**echo** "<th>Name</th>";

**echo** "<th>Price</th>";

**echo** "<th>Image</th>";

**echo** "</tr>";

$i = 0;

**while**($row = mysqli\_fetch\_array($result)){

**if**($i == 0):

**echo** "<p>Most Expensive Product: ".$row['name']."</p>";

**echo** "<p>Price: ".$row['price']."</p>";

**echo** "<p>Date Added: ".$row['date']."</p>";

**endif**;

**echo** "<tr>";

**echo** "<td>".$row['date']."</td>";

**echo** "<td>".$row['name']."</td>";

**echo** "<td>".$row['price']."</td>";

**echo** "<td><img src ='".UPLOAD\_PATH.$row['image']."' alt = 'Inventory Image'

height = '20px' width = '20px'/></td>";

**echo** "</tr>";

$i++;

}

**echo** "</table>";

?>

</body>

</htmL>

Code 30: orderinventory.php

#### File Validation

$\_FILES[‘image’][‘size’]

It gives the information about the file size

$\_FILES[‘image’][‘size’]

It gives the information about the file type

unlink() + @ = @unlink

It deletes a file from the web server. We can suppress error reporting with @ in case the file upload didn’t actually happen.

<?php

**require\_once**("common/databaseinfo.php");

$show\_form = **TRUE**;

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Connection Denied");

**if**(isset($\_POST['submit'])):

$name = $\_POST['item\_name'];

$price = $\_POST['price'];

$image\_name = $\_FILES['image']['name'];

$type = $\_FILES['image']['type'];

$size = $\_FILES['image']['size'];

$error = $\_FILES['image']['error'];

**if**(!**empty**($name) && !**empty**($price) && !**empty**($image\_name)):

**if**( ($type == "image/gif")

|| ($type == "image/jpeg")

|| ($type == "image/png")

|| ($type == "image/pjpeg") && ($size > 0) && $error == 0):

$show\_form = **FALSE**;

$target = UPLOAD\_PATH.$image\_name;

*// $date = date("Y-m-d h:i:s");*

$query = "INSERT INTO inventory VALUES(0, NOW(), '**$name**', '**$price**', '**$image\_name**')";

mysqli\_query($connection, $query) **or** **die**("Query Denied");

move\_uploaded\_file($\_FILES['image']['tmp\_name'], $target);

**echo** "Inventory Added Successfully";

**endif**;

**endif**;

@unlink($\_FILES['image']['tmp\_name']);

**endif**;

?>

<?php

**if**($show\_form):

?>

<!doctype html>

<html>

<head>

<title> Add Inventory </title>

<meta charset = "UTF-8"/>

</head>

<body>

<p> Add Inventory </p>

<form enctype = "multipart/form-data" method = "post"

action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<!-- <input type = "hidden" name = "MAX\_FILE\_SIZE" value = "32768"/> -->

<label for = "item\_name"> Item Name: </label><br/>

<input type = "text" name = "item\_name" id = "item\_name"/><br/>

<label for = "price"> Price: </label><br/>

<input type = "number" name = "price" id = "price" step = "0.01"/><br/>

<input type = "file" name = "image" id = "image"/><br/><br/>

<input type = "submit" value = "Add Item" name = "submit"/>

</body>

</html>

<?php

**endif**;

?>

Code 31: filevalidation.php

#### Admin Page

Web applications often include pages for public access, as well as admin pages that are only for site maintenance. The two types of web requests, GET and POST, control how we shuttle data between scripts.

#### Post and Get Requests

|  |  |
| --- | --- |
| **POST** | **GET** |
| It sends data to the server, after which the state of server usually changes in response to the data sent | Primarily to retrieve data from the server without affecting anything on the server |
| Works only in form tag | Works outside the form tag as well |
| POST requests can only be initiated through a form | The url of a script can be used to pass data as a GET request. |
| Data passed are not seen in the url | Data passed are seen in the url |
| INSERT, DELETE, UPDATE, SELECT | SELECT |

#### Limit

DELETE FROM inventory WHERE id = “$id” LIMIT 1

This puts safety, by creating a limit on the number of rows that can be deleted.

#### Remove Data Using GET and POST

<?php

**require**("common/databaseinfo.php");

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name);

$query = "SELECT \* FROM inventory";

$result = mysqli\_query($connection, $query);

?>

<!doctype html>

<html>

<head>

<title> Admin Remove Inventory </title>

</head>

<body>

<?php

**echo** "<table>";

**echo** "<tr>";

**echo** "<th>Date</th>";

**echo** "<th>Name</th>";

**echo** "<th>Price</th>";

**echo** "<th>Image</th>";

**echo** "<th>Remove</th>";

**echo** "</tr>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** "<td>".$row['date']."</td>";

**echo** "<td>".$row['name']."</td>";

**echo** "<td>".$row['price']."</td>";

**echo** "<td><img src ='".UPLOAD\_PATH.$row['image']."' alt = 'Inventory Image'

height = '20px' width = '20px'/></td>";

**echo** "<td><a href = 'adminremoveinventory.php?inventory\_id=".$row['inventory\_id']

."&amp;date=".$row['date']

."&amp;name=".$row['name']

."&amp;price=".$row['price']

."&amp;image=".$row['image']

."'>Remove</a></td>";

**echo**"</tr>";

}

**echo** "</table>";

mysqli\_close($connection);

?>

</form>

</body>

</html>

Code 32: admingetremove.php

<?php

$showconfirmform= **TRUE**;

**require**("common/databaseinfo.php");

**if**($showconfirmform):

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Remove Inventory Confirmation </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Error");

**if**( isset($\_GET['inventory\_id']) &&

isset($\_GET['name']) &&

isset($\_GET['price']) &&

isset($\_GET['image'])

):

**echo** "<h2> Are you sure? </h2>";

**echo** "<form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

$inventory\_id = $\_GET['inventory\_id'];

$name = $\_GET['name'];

$price = $\_GET['price'];

$image = $\_GET['image'];

$query = "SELECT \* FROM inventory WHERE inventory\_id = '**$inventory\_id**'";

$result = mysqli\_query($connection, $query) **or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

**echo** "Name: ".$row['name']."<br/>";

**echo** "Price: ".$row['price']."<br/>";

**echo** "<img src = '".UPLOAD\_PATH.$row['image']

."' alt = 'Inventory Image'"

."height='30px' width='30px'"

."/><br/>";

**echo** "<input type = 'hidden' name = 'inventory\_id' value = '**$inventory\_id**'/>";

**echo** "<input type = 'hidden' name = 'image' value = '".UPLOAD\_PATH.$row['image']."'/>";

**echo** "Yes<input type = 'radio' name = 'confirmation' value = 'yes'/>";

**echo** "No <input type = 'radio' name = 'confirmation' value = 'no'/><br/>";

**echo** "<input type = 'submit' name = 'submit' value = 'Confirm'/>";

}

**endif**;

**endif**;

?>

</form>

</body>

</html>

<?php

**if**(isset($\_POST['submit']) && ($\_POST['confirmation'] == "yes")):

$showconfirmform = **FALSE**;

$inventory\_id = $\_POST['inventory\_id'];

$query = "DELETE FROM inventory WHERE inventory\_id = '**$inventory\_id**'";

mysqli\_query($connection, $query) **or** **die**("Query Denied");

@unlink($\_POST['image']);

**echo** "Deleted Successfully!";

**endif**;

**if**(isset($\_POST['submit']) && ($\_POST['confirmation'] == "no")):

$showconfirmform = **FALSE**;

**echo** "<p><a href = 'admingetremove.php'>Home Page</a></p>";

**endif**;

?>

Code 33: adminremoveinventory.php

### Securing Application

HTTP authentication provides a simple way to secure to a page using PHP. HTTP is the hyper -text transfer protocol.

Username: $\_SERVER[‘PHP\_AUTH\_USER’]

Password: $\_SERVER[‘PHP\_AUTH\_PW’]

Browser and servers use headers quite often to communicate outside of the context of PHP, but PHP does allow us to send a header.

All web pages are delivered with the help of headers. HTTP authentication requires headers. Header control precisely how and what kind of information is passed back and forth between a web browser and web server.

#### HTTP Request

Get /index.php HTTP/1

It’s GET request for the page

Host: [www.google.com](http://www.google.com)

Most individual headers consist of a name/value pair separated by a colon

Checking header information using Chrome browser,

User-Agent: Mozilla/4.0…

Header specifies the browser that is doing the requesting

#### HTTP Response

HTTP/1.1 200 OK

Server’s JTTP response

Content-Type: text/html

This header tells the browser that the content is HTML code

Checking the header using the google chrome browser



Fig 15: Checking Header (CTRL+SHIFT+I) or F12

#### Headers

The header() function lets you create and send a header from a PHP script

Headers(‘Content-Type: text/html’);

// There should be no space before <?php tag

<?php

Server send this header to the browser for processing before attempting to send any of the HTML content in the page.

**Header(‘Content-Type: text/html’);**

Spaces inside of the <?php ?> tags aren’t a problem because they aren’t passed along to the browser.

?>

Two specific headers are required to request the authentication of a web page.

HTTP/1.1. 401 Unauthorized

This header lets the browser know that the user in not authorized to view the page

WWW-Authenticate: Basic realm=”store”

Basic realm is just a phrase used to uniquely identify this. Particular authentication-it appears in the authentication window.

exit()

An application has an opportunity to exit the script and display a custom denial message.

Headers can be used for other purposes too.

<?php

Header(’Location:http://www.addinventory.php’);

?>

The header is called a location header and redirects the current page to addinventory.php

<?php

Header(’Refresh: 5; url=http://www.addinventory.php’);

?>

This header is called a refresh header since it refreshes a page after a period of time.

<?php

Header(’Content-Type: text-plain’);

?>

Headers must be the very first thing sent to the browser in a PHP file. It is because the headers must be sent before any content, it is extremely important to not allow even a single space to appear outside of PHP code before calling the header() function in a PHP script.

Basic Realm defines a security zone that is protected by a particular username and password. Once the user name and password have been successfully entered for a given realm, the browser will remember it and not continue to display the authentication window for subsequent authentication headers in the same realm. In other words, realms allow a browser to remember that we have met the security requirements of a given collection of pages—just specify the same realm for the authentication headers in the pages.

<?php

$username = "photon";

$password = "khan";

**if**(

!isset($\_SERVER['PHP\_AUTH\_USER']) ||

!isset($\_SERVER['PHP\_AUTH\_PW']) ||

($\_SERVER['PHP\_AUTH\_USER'] != $username) ||

($\_SERVER['PHP\_AUTH\_PW'] != $password)

):

header('HTTP/1.1 401 Unauthorized');

header('WWW-Authenticate: Basic realm="Store"');

**exit**('<h2> You must enter a valid username and password');

**endif**;

?>

Code 34: header.php

#### Human Moderation

Human moderation is an excellent way to improve the integrity of user-submitted content.

Adding approval link beside the remove

ALTER TABLE inventory ADD COLUMN approved TINYINT;

Alternatively, ALTER TABLE inventory ADD COLUMN approved ENUM(‘yes’, ‘no’)

UPDATE inventory SET approved = 1 WHERE inventory\_id = '$inventory\_id'

Use this to update the data in the table

#### SQL Injection

Tricking MySQL with comments. A double hyphen (--) is used in SQL to comment out the remainder of a line of SQL code. To make it work, it is double hyphen with a space (-- ), everything after it is ignored.

SQL code:

INSERT INTO inventory VALUES(0, NOW(), ‘Hacker’, ‘10000000’, ‘hacker.png’, 1) – hacker.png

1 is used to approve and everything after – is commented.

Form fields are security weak point for web applications because they allow users to enter data.

Dangerous characters are any characters that could possibly change the nature of any SQL-query, such as commas, quotes or – comment characters. Even the spaces at the end of a piece of data can prove harmful. SQL injections can be prevented by properly processing form data.

trim()

Eliminates Leading or trailing spaces

Ex: $name = trim($\_POST[‘name’]);

mysqli\_real\_escape\_string($connection, trim($\_POST[‘name]));

It converts dangerous characters into an escaped format that won’t adversely affect SQL queries.

The id and approved columns can then be allowed to default to AUTO\_INCREMENT and 0, respectively.

<!doctype html>

<html>

<head>

<title> Inventory Index </title>

<meta charset = "UTF-8"/>

</head>

<body>

<h1> Inventory **List** </h1>

<p><a href = "addinventory.php">Add Item</a></p>

<p> **List** of items are given below</p>

<?php

**require\_once**("common/databaseinfo.php");

$connection = mysqli\_connect($server\_name, $user\_name,

$password, $db\_name)

**or** **die**("Could not conenct to the database");

$query = "SELECT \* FROM inventory";

$result = mysqli\_query($connection, $query);

**echo** "<table>";

**echo** "<tr>";

**echo** "<th>Date</th>";

**echo** "<th>Name</th>";

**echo** "<th>Price</th>";

**echo** "<th>Image</th>";

**echo** "</tr>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**if**($row['approved'] == 1):

**echo** "<td>".$row['date']."</td>";

**echo** "<td>".$row['name']."</td>";

**echo** "<td>".$row['price']."</td>";

**echo** "<td><img src ='".UPLOAD\_PATH.$row['image']."' alt = 'Inventory Image'

height = '20px' width = '20px'/></td>";

**endif**;

**echo** "</tr>";

}

**echo** "</table>";

?>

</body>

</html>

Code 35: inventoryindex.php

<?php

**require\_once**("common/databaseinfo.php");

$show\_form = **TRUE**;

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Connection Denied");

**if**(isset($\_POST['submit'])):

$name = mysqli\_real\_escape\_string($connection, trim($\_POST['item\_name']));

$price = mysqli\_real\_escape\_string($connection, trim($\_POST['price']));

$image\_name = mysqli\_real\_escape\_string($connection, trim($\_FILES['image']['name']));

**if**(!**empty**($name) && !**empty**(is\_numeric($price)) && !**empty**($image\_name)):

$show\_form = **FALSE**;

$target = UPLOAD\_PATH.$image\_name;

*// $date = date("Y-m-d h:i:s");*

$query = "INSERT INTO inventory(date, name, price, image)

VALUES(NOW(), '**$name**', '**$price**', '**$image\_name**')";

mysqli\_query($connection, $query) **or** **die**("Query Denied");

move\_uploaded\_file($\_FILES['image']['tmp\_name'], $target);

**echo** "Inventory Added Successfully";

**endif**;

**endif**;

?>

<?php

**if**($show\_form):

?>

<!doctype html>

<html>

<head>

<title> Add Inventory </title>

<meta charset = "UTF-8"/>

</head>

<body>

<p> Add Inventory </p>

<form enctype = "multipart/form-data" method = "post"

action = "<?php **echo** $\_SERVER['PHP\_SELF'];?>">

<!-- <input type = "hidden" name = "MAX\_FILE\_SIZE" value = "32768"/> -->

<label for = "item\_name"> Item Name: </label><br/>

<input type = "text" name = "item\_name" id = "item\_name"/><br/>

<label for = "price"> Price: </label><br/>

<input type = "number" name = "price" id = "price" step = "0.01"/><br/>

<input type = "file" name = "image" id = "image"/><br/><br/>

<input type = "submit" value = "Add Item" name = "submit"/>

</body>

</html>

<?php

**endif**;

?>

Code 36: addinventory.php

<?php **require\_once**("common/header.php"); ?>

<?php

**require\_once**("common/databaseinfo.php");

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name);

$query = "SELECT \* FROM inventory";

$result = mysqli\_query($connection, $query);

?>

<!doctype html>

<html>

<head>

<title> Admin Remove Inventory </title>

</head>

<body>

<?php

**echo** "<table>";

**echo** "<tr>";

**echo** "<th>Date</th>";

**echo** "<th>Name</th>";

**echo** "<th>Price</th>";

**echo** "<th>Image</th>";

**echo** "<th>Remove</th>";

**echo** "<th>Status</th>";

**echo** "</tr>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** "<td>".$row['date']."</td>";

**echo** "<td>".$row['name']."</td>";

**echo** "<td>".$row['price']."</td>";

**echo** "<td><img src ='".UPLOAD\_PATH.$row['image']."' alt = 'Inventory Image'

height = '20px' width = '20px'/></td>";

**echo** "<td><a href = 'adminremoveinventory.php?inventory\_id=".$row['inventory\_id']

."&amp;date=".$row['date']

."&amp;name=".$row['name']

."&amp;price=".$row['price']

."&amp;image=".$row['image']

."'>Remove</a></td>";

**if**($row['approved'] == 0):

**echo** "<td><a href = 'approve.php?inventory\_id=".$row['inventory\_id']."'>Approve</a></td>";

**endif**;

**echo**"</tr>";

}

**echo** "</table>";

mysqli\_close($connection);

?>

</form>

</body>

</html>

Code 37: admingetremove.php

<?php **require\_once**("common/header.php");?>

<?php

$showconfirmform= **TRUE**;

**require**("common/databaseinfo.php");

**if**($showconfirmform):

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Approve Inventory </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Error");

**if**(isset($\_GET['inventory\_id'])):

**echo** "<h2> Are you sure? </h2>";

**echo** "<form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

$inventory\_id = $\_GET['inventory\_id'];

$query = "SELECT \* FROM inventory WHERE inventory\_id = '**$inventory\_id**'";

$result = mysqli\_query($connection, $query) **or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

**echo** "Name: ".$row['name']."<br/>";

**echo** "Price: ".$row['price']."<br/>";

**echo** "<img src = '".UPLOAD\_PATH.$row['image']

."' alt = 'Inventory Image'"

."height='30px' width='30px'"

."/><br/>";

**echo** "<input type = 'hidden' name = 'inventory\_id' value = '**$inventory\_id**'/>";

**echo** "<input type = 'hidden' name = 'image' value = '".UPLOAD\_PATH.$row['image']."'/>";

**echo** "Yes<input type = 'radio' name = 'confirmation' value = 'yes'/>";

**echo** "No <input type = 'radio' name = 'confirmation' value = 'no'/><br/>";

**echo** "<input type = 'submit' name = 'submit' value = 'Confirm'/>";

}

**endif**;

**endif**;

?>

</form>

</body>

</html>

<?php

**if**(isset($\_POST['submit']) && ($\_POST['confirmation'] == "yes")):

$showconfirmform = **FALSE**;

$inventory\_id = $\_POST['inventory\_id'];

$query = "UPDATE inventory SET approved = 1 WHERE inventory\_id = '**$inventory\_id**'";

mysqli\_query($connection, $query) **or** **die**("Query Denied");

**echo** "Approved Successfully!";

**echo** "<p><a href = 'admingetremove.php'>Home Page</a></p>";

**endif**;

**if**(isset($\_POST['submit']) && ($\_POST['confirmation'] == "no")):

$showconfirmform = **FALSE**;

**echo** "<p><a href = 'admingetremove.php'>Home Page</a></p>";

**endif**;

?>

Code 38: approve.php

### Personal Web Application

Personal web applications thrive on personal information, which requires users to be able to access an application on a personal level.

User Log-ins allow web applications to get personal with users. An application log-in requires a user interface for entering the username and password. The asterisks displayed in a password form field simple provide visual security, preventing someone from looking over our should as we enter the password. When the form is submitted, the password itself is submitted, not the asterisks. That’s why it’s important for the password to be encrypted before inserting it into the database.

MySQL, SHA() function encrypts a piece of text into a unique 40-character code.

The SHA() function provides one-way encryption-we can’t decrypt data that has been encrypted.

SHA() functions stands for **Secure Hash Algorithm**. A “**hash**” is a programming term that refers to a unique, fixed-length string that uniquely represents a string of text. In the case of SHA(), the hash is the 40-character hexadecimal encrypted string of text, which uniquely represents the original password.

Other ways to encrypt data is using **MD5()** in my **MySQL**

**SHA()** is considered a little secure more than **MD5()**

PHP also offers a equivalent function (**sha1()** and **md5()**)

**HTTP authentication** is intended to be carried out once for a **given page or collection of pages**—it’s only reset when the **browser is shut down**. In other words, a user is never “logged out” of an HTTP authenticated web page until the browser is shut down or the user manually clears the HTTP authenticated session.

<?php

**require\_once**("common/database.php");

**require\_once**("common/constants.php");

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die** ("Server Denied");

$showform = **TRUE**;

**if**(isset($\_POST['submit'])):

**if**(

!**empty**($\_POST['username']) &&

!**empty**($\_POST['password']) &&

!**empty**($\_POST['first\_name']) &&

!**empty**($\_POST['last\_name']) &&

!**empty**($\_POST['gender']) &&

!**empty**($\_POST['date']) &&

!**empty**($\_POST['city']) &&

!**empty**($\_POST['state']) &&

!**empty**($\_FILES['picture']) &&

$\_POST['password'] == $\_POST['password\_two']

):

$showform = **FALSE**;

$username = isset($\_POST['username'])? mysqli\_real\_escape\_string($connection, trim($\_POST['username'])): **NULL**;

$password = isset($\_POST['password'])? mysqli\_real\_escape\_string($connection, trim($\_POST['password'])): **NULL**;

$first\_name = isset($\_POST['first\_name'])? mysqli\_real\_escape\_string($connection, trim($\_POST['first\_name'])): **NULL**;

$last\_name = isset($\_POST['last\_name'])? mysqli\_real\_escape\_string($connection, trim($\_POST['last\_name'])): **NULL**;

$gender = isset($\_POST['gender'])? mysqli\_real\_escape\_string($connection, trim($\_POST['gender'])): **NULL**;

$date = isset($\_POST['date'])? mysqli\_real\_escape\_string($connection, trim($\_POST['date'])): **NULL**;

$city = isset($\_POST['city'])? mysqli\_real\_escape\_string($connection, trim($\_POST['city'])): **NULL**;

$state = isset($\_POST['state'])? mysqli\_real\_escape\_string($connection, trim($\_POST['state'])): **NULL**;

$picture = isset($\_FILES['picture'])? mysqli\_real\_escape\_string($connection, trim($\_FILES['picture']['name'])): **NULL**;

$query = "INSERT INTO mismatch\_user(username, password, join\_date, first\_name, last\_name, gender, birth\_date, city, state, picture)

VALUES('**$username**', SHA('**$password**'), NOW(), '**$first\_name**', '**$last\_name**', '**$gender**', '**$date**', '**$city**', '**$state**', '**$picture**')";

mysqli\_query($connection, $query);

$destination = UPLOAD\_PATH.$\_FILES['picture']['name'];

move\_uploaded\_file($\_FILES['picture']['tmp\_name'], $destination);

**echo** "User Successfully Added";

**endif**;

**endif**;

**if**($showform):

**echo** '<!doctype html>';

**echo** ' <html>';

**echo** ' <head>';

**echo** ' <title> Add Users </title>';

**echo** ' <meta charset = "UTF-8"/>';

**echo** ' </head>';

**echo** ' <body>';

**echo** " <form enctype = 'multipart/form-data' method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** ' <label for = "username">Username</label><br/>';

**echo** ' <input type = "text" name = "username"/><br/>';

**echo** ' <label for = "password">Password</label><br/>';

**echo** ' <input type = "password" name = "password"/><br/>';

**echo** ' <label for = "password\_two">Retype Password</label><br/>';

**echo** ' <input type = "password" name = "password\_two"/><br/>';

**echo** ' <label for = "first\_name">First Name</label><br/>';

**echo** ' <input type = "text" name = "first\_name"/><br/>';

**echo** ' <label for = "last\_name">Last Name</label><br/>';

**echo** ' <input type = "text" name = "last\_name"/><br/>';

**echo** ' <label for = "gender">Gender</label><br/>';

**echo** ' Male<input type = "radio" name = "gender" value = "male"/>';

**echo** ' Female<input type = "radio" name = "gender" value = "female"/><br/>';

**echo** ' <label for = "date">Date</label><br/>';

**echo** ' <input type = "date" name = "date"/><br/>';

**echo** ' <label for = "city">City</label><br/>';

**echo** ' <input type = "text" name = "city"/><br/>';

**echo** ' <label for = "state">State</label><br/>';

**echo** ' <input type = "text" name = "state"/><br/>';

**echo** ' <label for = "picture">Picture</label><br/>';

**echo** ' <input type = "file" name = "picture"/><br/>';

**echo** ' <input type = "submit" name = "submit" value = "Submit"/><br/>';

**echo** ' </form>';

**echo** ' </body>';

**echo** ' </html>';

**endif**;

?>

Code 38: signup.php

<?php

**require\_once**("database.php");

**if**(!isset($\_SERVER['PHP\_AUTH\_USER']) || !isset($\_SERVER['PHP\_AUTH\_PW'])):

header('HTTP/1.1 Unauthorized');

header('WWW-Authenticate: Basic realm="Mismatch"');

**exit**('<h3> Mismatch </h3> Sorry, you must enter your username and

this password to log in and access');

**endif**;

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die** ("Server Denied");

$user\_username = mysqli\_real\_escape\_string($connection, trim($\_SERVER['PHP\_AUTH\_USER']));

$user\_password = mysqli\_real\_escape\_string($connection, trim($\_SERVER['PHP\_AUTH\_PW']));

$query = "SELECT user\_id, username FROM mismatch\_user WHERE username = '**$user\_username**' AND password = SHA('**$user\_password**')";

$data = mysqli\_query($connection, $query) **or** **die** ("Query Denied");

**if**(mysqli\_num\_rows($data) == 1){

$row = mysqli\_fetch\_array($data);

$user\_id = $row['user\_id'];

$username = $row['username'];

}

**else**{

header('HTTP/1.1 Unauthorized');

header('WWW-Authenticate: Basic realm="Mismatch"');

**exit**('<h3> Mismatch </h3> Sorry, you must enter your username and

this password to log in and access');

}

?>

Code 39: authentication.php

HTTP authentication stores data persistently on the client but doesn’t allow us to delete it when we are done.

mysqli\_num\_rows($data)

Checks the number of rows in $data

#### Cookies

Cookies allow us to persistently store **small pieces of data on the client** that can outlive any single script and can be deleted at will. Cookie data is stored on the user’s computer by their web browser. A cookie stores a single piece of data under a unique name, much like a variable in PHP. A cookie can have an expiration date. When this expiration date arrives, the cookie is destroyed. PHP setcookie() function allows us to store data in cookies. The superglobal is called $\_COOKIE.

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

setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/");

The cookie will expire after 30 days (86400 \* 30). The "/" means that the cookie is available in entire website

Ex: setcookie(‘username’, ‘sidneyk’);

Ex: $\_COOKIE[‘username’];

setcookie() function also accepts an optional third argument that sets the expiration date of the cookie, which is the date upon which the cookie is automatically deleted.

To delete cookie, setcookie(‘username’, ‘marylee’, time() - (60 \* 60);

To delete a cookie, just set its expiration date to a time in the past.

dirname($\_SERVER['PHP\_SELF'])

/education/Php/7. Personal Web Application

$\_SERVER['HTTP\_HOST']

Localhost

<?php

define("UPLOAD\_PATH","images/");

?>

Code 40: constant.php

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "mismatch";

?>

Code 41: database.php

<?php

**require\_once**("common/constants.php");

**require\_once**("common/database.php");

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Denied");

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Home Page </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <h1> Mismatch - Leave the Love Ones Behind </h1>";

**if**(!isset($\_COOKIE['username'])){

**echo** "<p><a href = 'login.php'>Log In</a>/<a href = 'signup.php'>Sign Up</a></p>";

}

**else**{

**echo** "<p><a href = 'view.php'>View</a>/<a href = 'edit.php'>Edit</a></p>";

**echo** "<p><a href = 'logout.php'>Log Out</a></p>";

}

**echo** " <table>";

**echo** " <tr>";

**echo** " <th> Latest Members</th>";

**echo** " <th></th>";

**echo** " </tr>";

$query = "SELECT first\_name, last\_name, picture FROM mismatch\_user ORDER BY birth\_date DESC LIMIT 5";

$result = mysqli\_query($connection, $query) **or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** " <td>".$row['first\_name']." ".$row['last\_name']."</td>";

**echo** " <td><img src = '".UPLOAD\_PATH.$row['picture']."' alt = 'profile photo'"

."height = '50px' width = '50px'"

."</td>";

**echo** "</tr>";

}

**echo** " </table>";

**echo** " </body>";

**echo** " </html>";

?>

Code 41: home.php

<?php

**require\_once**("common/database.php");

**require\_once**("common/constants.php");

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die** ("Server Denied");

$showform = **TRUE**;

**if**(isset($\_POST['submit'])):

**if**(

!**empty**($\_POST['username']) &&

!**empty**($\_POST['password']) &&

!**empty**($\_POST['first\_name']) &&

!**empty**($\_POST['last\_name']) &&

!**empty**($\_POST['gender']) &&

!**empty**($\_POST['date']) &&

!**empty**($\_POST['city']) &&

!**empty**($\_POST['state']) &&

!**empty**($\_FILES['picture']) &&

$\_POST['password'] == $\_POST['password\_two']

):

$showform = **FALSE**;

$username = isset($\_POST['username'])? mysqli\_real\_escape\_string($connection, trim($\_POST['username'])): **NULL**;

$password = isset($\_POST['password'])? mysqli\_real\_escape\_string($connection, trim($\_POST['password'])): **NULL**;

$first\_name = isset($\_POST['first\_name'])? mysqli\_real\_escape\_string($connection, trim($\_POST['first\_name'])): **NULL**;

$last\_name = isset($\_POST['last\_name'])? mysqli\_real\_escape\_string($connection, trim($\_POST['last\_name'])): **NULL**;

$gender = isset($\_POST['gender'])? mysqli\_real\_escape\_string($connection, trim($\_POST['gender'])): **NULL**;

$date = isset($\_POST['date'])? mysqli\_real\_escape\_string($connection, trim($\_POST['date'])): **NULL**;

$city = isset($\_POST['city'])? mysqli\_real\_escape\_string($connection, trim($\_POST['city'])): **NULL**;

$state = isset($\_POST['state'])? mysqli\_real\_escape\_string($connection, trim($\_POST['state'])): **NULL**;

$picture = isset($\_FILES['picture'])? mysqli\_real\_escape\_string($connection, trim($\_FILES['picture']['name'])): **NULL**;

$query = "INSERT INTO mismatch\_user(username, password, join\_date, first\_name, last\_name, gender, birth\_date, city, state, picture)

VALUES('**$username**', SHA('**$password**'), NOW(), '**$first\_name**', '**$last\_name**', '**$gender**', '**$date**', '**$city**', '**$state**', '**$picture**')";

mysqli\_query($connection, $query);

$destination = UPLOAD\_PATH.$\_FILES['picture']['name'];

move\_uploaded\_file($\_FILES['picture']['tmp\_name'], $destination);

**echo** "User Successfully Added";

**endif**;

**endif**;

**if**($showform):

**echo** '<!doctype html>';

**echo** ' <html>';

**echo** ' <head>';

**echo** ' <title> Add Users </title>';

**echo** ' <meta charset = "UTF-8"/>';

**echo** ' </head>';

**echo** ' <body>';

**echo** " <form enctype = 'multipart/form-data' method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** ' <label for = "username">Username</label><br/>';

**echo** ' <input type = "text" name = "username"/><br/>';

**echo** ' <label for = "password">Password</label><br/>';

**echo** ' <input type = "password" name = "password"/><br/>';

**echo** ' <label for = "password\_two">Retype Password</label><br/>';

**echo** ' <input type = "password" name = "password\_two"/><br/>';

**echo** ' <label for = "first\_name">First Name</label><br/>';

**echo** ' <input type = "text" name = "first\_name"/><br/>';

**echo** ' <label for = "last\_name">Last Name</label><br/>';

**echo** ' <input type = "text" name = "last\_name"/><br/>';

**echo** ' <label for = "gender">Gender</label><br/>';

**echo** ' Male<input type = "radio" name = "gender" value = "male"/>';

**echo** ' Female<input type = "radio" name = "gender" value = "female"/><br/>';

**echo** ' <label for = "date">Date</label><br/>';

**echo** ' <input type = "date" name = "date"/><br/>';

**echo** ' <label for = "city">City</label><br/>';

**echo** ' <input type = "text" name = "city"/><br/>';

**echo** ' <label for = "state">State</label><br/>';

**echo** ' <input type = "text" name = "state"/><br/>';

**echo** ' <label for = "picture">Picture</label><br/>';

**echo** ' <input type = "file" name = "picture"/><br/>';

**echo** ' <input type = "submit" name = "submit" value = "Submit"/><br/>';

**echo** ' </form>';

**echo** ' </body>';

**echo** ' </html>';

**endif**;

?>

Code 42: signup.php

<?php

**require\_once**("common/database.php");

**if**(!isset($\_COOKIE['user\_id'])):

**if**(isset($\_POST['submit'])):

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die** ("Server Denied");

$user\_username = mysqli\_real\_escape\_string($connection, trim($\_POST['username']));

$user\_password = mysqli\_real\_escape\_string($connection, trim($\_POST['password']));

**if**(!**empty**($user\_username) && !**empty**($user\_password)):

$query = "SELECT user\_id, username FROM mismatch\_user WHERE username = '**$user\_username**' AND password = SHA('**$user\_password**')";

$data = mysqli\_query($connection, $query) **or** **die** ("Query Denied");

**if**(mysqli\_num\_rows($data) == 1):

$row = mysqli\_fetch\_array($data);

setcookie('user\_id',$row['user\_id'],time() + (86400 \* 30), "/");

setcookie('username',$row['username'],time() + (86400 \* 30), "/");

$home\_url = "http://".$\_SERVER['HTTP\_HOST'].dirname($\_SERVER['PHP\_SELF'])."/home.php";

**echo** "Successfully Logged In";

header("Location: ".$home\_url);

**endif**;

**endif**;

**endif**;

**endif**;

**if**(**empty**($\_COOKIE['user\_id'])){

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title>Log in</title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <h3> Log In </h3>";

**echo** " <form method = 'post' action = ''>";

**echo** " <label for = 'username'>Username</label><br/>";

**echo** " <input type = 'text' name = 'username'><br/>";

**echo** " <label for = 'password'> Password </label><br/>";

**echo** " <input type = 'password' name = 'password'/><br/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Confirm'/>";

**echo** " </form>";

**echo** " </body>";

**echo** " </html>";

}

**else**{

**echo** "You are logged in as ".$\_COOKIE['username'];

}

?>

Code 42: login.php

<?php

**if**(

isset($\_COOKIE['username']) && !**empty**($\_COOKIE['username']) &&

isset($\_COOKIE['user\_id']) && !**empty**($\_COOKIE['user\_id'])

):

**echo** "<p>".$\_COOKIE['username']." Successfully logged out</p>";

setcookie('username',' ', time() - 3600, "/");

setcookie('user\_id',' ', time() - 3600, "/");

$home\_url = "http://".$\_SERVER['HTTP\_HOST'].dirname($\_SERVER['PHP\_SELF'])."/home.php";

header("Location: ".$home\_url);

**endif**;

?>

Code 43: logout.php

<?php

**require\_once**("common/constants.php");

**require\_once**("common/database.php");

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Denied");

**if**(isset($\_COOKIE['username']) && isset($\_COOKIE['user\_id'])):

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Home Page </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <h1> Mismatch - Leave the Love Ones Behind </h1>";

**echo** " <p> You are in logged in as ".$\_COOKIE['username']

.",<a href = 'logout.php'> Log Out</a>"

."</p>";

$query = "SELECT \* FROM mismatch\_user WHERE user\_id =".$\_COOKIE['user\_id'];

$result = mysqli\_query($connection, $query) **or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

**echo** " <table>";

**echo** " <tr>";

**echo** " <th></th>";

**echo** " <th></th>";

**echo** " </tr>";

**echo** " <tr>";

**echo** " <td>";

**echo** " <p> Username: </p>";

**echo** " <p> First Name: </p>";

**echo** " <p> Last Name: </p>";

**echo** " <p> Join Date: </p>";

**echo** " <p> Birth Date: </p>";

**echo** " <p> City: </p>";

**echo** " <p> State: </p>";

**echo** " </td>";

**echo** " <td>";

**echo** "<p>".$row['username']."</p>";

**echo** "<p>".$row['first\_name']."</p>";

**echo** "<p>".$row['last\_name']."</p>";

**echo** "<p>".$row['join\_date']."</p>";

**echo** "<p>".$row['birth\_date']."</p>";

**echo** "<p>".$row['city']."</p>";

**echo** "<p>".$row['state']."</p>";

**echo** " </td>";

**echo** " </tr>";

**echo** " </table>";

**echo** " <p>Photo</p>";

**echo** " <img src = '".UPLOAD\_PATH.$row['picture']."' alt = 'profile photo'"."height = '100px' width = '100px'/>";

}

**echo** "<p> Would you like to <a href = 'edit.php'>edit</a> your profile?</p>";

**echo** "<p> <a href = 'home.php'>Home</a></p>";

**echo** " </body>";

**echo** " </html>";

**endif**;

?>

Code 44: view.php

#### Sessions

Sessions allow us to persistently store **small pieces of data on the server**, independently of the client. Since, session data is stored on the server, it is more secure and more reliable than data stored in cookies. Session variables are automatically destroyed as soon as a session ends (closing the browser), which usually coincides with the user shutting down the browser. Session can store larger amounts of data than a cookie.

|  |  |
| --- | --- |
| **COOKIE** | **SESSION** |
| Data saved on the client | Data saved on the server |
| setcookie() to store data | $\_SESSION assign to the super variable |
| Smaller amount of data | Larger amount of data |
| Less Secure | More Secure |
| setcookie() does the work | Session ID to share access |
| setcookie() closes to negative time | $\_SESSION = array() |
| Has an expiration date to destroy | Short lived, browser closes destroys |

To start a session: session\_start();

The PHP session\_start() starts a session and allows us to begin storing data in session variables.

To close a session: session\_destroy();

The session ID is used behind the scenes to allow multiple pages to share access to session data. The session ID isn’t destroyed until the session is closed, which happens either when the browser is closed or when we call the session\_destroy() function. This doesn’t destroy the session variables.

Unlike cookies, session variables don’t require any kind of special function to set them—we just assign a value to the $\_SESSION superglobal.

$\_SESSION = array();

This code kills all of the session variables in the current session.

$\_SESSION is the superglobal

Ex: $\_SESSION(‘username’) = ‘marylee’;

$\_COOKIE[session\_name()]

If a session is using a cookie to help remember the session ID, then the ID is stored in a cookie named after session

setcookie($\_COOKIE[session\_name(), ‘’, time() - 3600, ‘/’];

Destroy the session cookie by settings its expiration to an hour in the past.

SESSION without COOKIES may not work if our PHP settings in php.ini aren’t configured properly on the server. Therefore, we can pass session ID among different pages by appending to the URL of each page, which takes place automatically if the **session.use\_trans\_id** setting is **set to 1** (true) in the **php.ini** file on the server. We can manually append the session ID to the URL of session pages by

<a href = “view.php?<?php echo SID; ?>”>View Your Profile</a>

Session and Cookie are not stored in database because it is ideally suited for holding permanent data.

Therefore, we can use the combination of cookies and session to increase the persistence of the website.

<?php

**require\_once**("common/constants.php");

**require\_once**("common/database.php");

session\_start();

**if**(!isset($\_SESSION['user\_id']) || !isset($\_SESSION['username'])):

**if**(isset($\_COOKIE['user\_id']) && isset($\_COOKIE['username'])):

$\_SESSION['user\_id'] = $\_COOKIE['user\_id'];

$\_SESSION['username'] = $\_COOKIE['username'];

**endif**;

**endif**;

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Denied");

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Home Page </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <h1> Mismatch - Leave the Love Ones Behind </h1>";

**if**(!isset($\_COOKIE['username'])){

**echo** "<p><a href = 'loginsession.php'>Log In</a>/<a href = 'signup.php'>Sign Up</a></p>";

}

**else**{

**echo** "<p><a href = 'view.php'>View</a>/<a href = 'edit.php'>Edit</a></p>";

**echo** "<p><a href = 'logoutsession.php'>Log Out</a></p>";

}

**echo** " <table>";

**echo** " <tr>";

**echo** " <th> Latest Members</th>";

**echo** " <th></th>";

**echo** " </tr>";

$query = "SELECT first\_name, last\_name, picture FROM mismatch\_user ORDER BY birth\_date DESC LIMIT 5";

$result = mysqli\_query($connection, $query) **or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** " <td>".$row['first\_name']." ".$row['last\_name']."</td>";

**echo** " <td><img src = '".UPLOAD\_PATH.$row['picture']."' alt = 'profile photo'"

."height = '50px' width = '50px'"

."</td>";

**echo** "</tr>";

}

**echo** " </table>";

**echo** " </body>";

**echo** " </html>";

?>

Code 44: homesession.php

<?php

session\_start();

**if**(

isset($\_SESSION['username']) && !**empty**($\_SESSION['username']) &&

isset($\_SESSION['user\_id']) && !**empty**($\_SESSION['user\_id'])

):

$\_SESSION = **array**();

**if**(isset($\_COOKIE[session\_name()])):

setcookie(session\_name(), '', time() - 3600, '/');

**endif**;

session\_destroy();

**endif**;

**if**(

isset($\_COOKIE['username']) && !**empty**($\_COOKIE['username']) &&

isset($\_COOKIE['user\_id']) && !**empty**($\_COOKIE['user\_id'])

):

setcookie('username',' ', time() - 3600, "/");

setcookie('user\_id',' ', time() - 3600, "/");

$home\_url = "http://".$\_SERVER['HTTP\_HOST'].dirname($\_SERVER['PHP\_SELF'])."/homesession.php";

header("Location: ".$home\_url);

**endif**;

?>

Code 44: logoutsession.php

<?php

**require\_once**("common/database.php");

session\_start();

**if**(!isset($\_COOKIE['user\_id'])):

**if**(isset($\_POST['submit'])):

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die** ("Server Denied");

$user\_username = mysqli\_real\_escape\_string($connection, trim($\_POST['username']));

$user\_password = mysqli\_real\_escape\_string($connection, trim($\_POST['password']));

**if**(!**empty**($user\_username) && !**empty**($user\_password)):

$query = "SELECT user\_id, username FROM mismatch\_user WHERE username = '**$user\_username**' AND password = SHA('**$user\_password**')";

$data = mysqli\_query($connection, $query) **or** **die** ("Query Denied");

**if**(mysqli\_num\_rows($data) == 1):

$row = mysqli\_fetch\_array($data);

$\_SESSION['user\_id'] = $row['user\_id'];

$\_SESSION['username'] = $row['username'];

setcookie('user\_id',$row['user\_id'],time() + (86400 \* 30), "/");

setcookie('username',$row['username'],time() + (86400 \* 30), "/");

$home\_url = "http://".$\_SERVER['HTTP\_HOST'].dirname($\_SERVER['PHP\_SELF'])."/homesession.php";

**echo** "Successfully Logged In";

header("Location: ".$home\_url);

**endif**;

**endif**;

**endif**;

**endif**;

**if**(**empty**($\_COOKIE['user\_id'])){

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title>Log in</title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <h3> Log In </h3>";

**echo** " <form method = 'post' action = ''>";

**echo** " <label for = 'username'>Username</label><br/>";

**echo** " <input type = 'text' name = 'username'><br/>";

**echo** " <label for = 'password'> Password </label><br/>";

**echo** " <input type = 'password' name = 'password'/><br/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Confirm'/>";

**echo** " </form>";

**echo** " </body>";

**echo** " </html>";

}

**else**{

**echo** "You are logged in as ".$\_COOKIE['username'];

}

?>

Code 44: loginsession.php

#### Templates

Templates allow a PHP application to be build out of reusable script components. Typical parts of templates are headers, navigation menu, footer.

Headers appears at the top of every page, and displays the application title as well as a page-specific title. Footer provides content along the bottom of every page, which includes a copyright notice.

<?php

**echo** "<p>Copyright &copy;".date("Y")." Jack Enterprise Inc.";

?>

Code 44: pagefooter.php

### Control Data

Schema

A description of the data (the tables and columns) in our database, along with any other related objects and the way they all connect is known as a schema. Creating a diagram of a table lets us keep the design of the table separate from the data that’s inside of it.

A **foreign key** is a column in a table **references the primary key** of another table.

Binding together tables with primary keys and foreign keys allows us to connect data between them in a consistent manner. We can even structure our database so that primary keys and their respective foreign keys are required to match up. This is known as **referential integrity;** all keys must be valid.



Fig 16: One-to-One Schema Design

One-to-One: Exactly one row of a parent table is related to one row of a child table.



Fig 17: One-to-Many Schema Design

One-to-Many: Exactly one row of a parent table is related to multiple rows of a child table



Fig 18: Many-to-Many Schema Design

Many-to-Many: Multiple rows of a parent table are related to multiple rows of child table

Response table is as **junction table,** by serving as a convenient go-between for the users and topics.

If we start with a well-designed database, every other piece of the application puzzle becomes that much easier to build and assemble.

**array\_push()**

It tacks a new element onto the end of an array, causing the array to grow by one

<?php  
 $a=array("a"=>"red","b"=>"green");  
 array\_push($a,"blue","yellow");  
 print\_r($a);  
?>

**Solution:** Array ([a] => red [b] => green [0] => blue [1] => yellow)

**Foreach**

foreach($array as $key => $value){  
    // Code to be executed  
}

<?php

$superhero = **array**(

"name" => "Peter Parker",

"email" => "peterparker@mail.com",

"age" => 18

);

*// Loop through superhero array*

**foreach**($superhero **as** $key => $value){

**echo** $key . " : " . $value . "<br>";

}

?>

**Solution**

name : Peter Parker  
email : peterparker@mail.com  
age : 18

Code 45: foreach(key,value).php

**Data driven forms rely on data in a MySQL database to generate HTML form fields.**

The ternary ? : operator can be used to code if-else statements in a compact form



<?php

**require\_once**("common/sessionstarter.php");

**require\_once**("common/database.php");

*// Check if the cookier user id exists or not*

$cookie\_user\_id = $\_COOKIE['user\_id'];

**if**(!isset($cookie\_user\_id)):

$url = "http://".$\_SERVER['HTTP\_HOST'].dirname($\_SERVER['PHP\_SELF'])."/home.php";

header("Location: **$url**");

**endif**;

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die**("Server Denied");

*/\*\**

*\**

*\* Filling up the mismatch\_response with topic\_ids w.r.t user\_id*

*\**

*\*/*

*//Check whether the mismatch\_response has any topic\_id, response\_id, response w.r.t user\_id in the mismatch\_response*

$query = "SELECT \* FROM mismatch\_response WHERE user\_id = '**$cookie\_user\_id**'";

$result = mysqli\_query($connection, $query) **or** **die**("User in Response Table Query Denied");

$topicIDs = **array**();

*//If the mismatch\_response has no data, create topic\_id w.r.t user\_id*

**if**(mysqli\_num\_rows($result) == 0):

*//Selecting all the topic\_id from the topic table*

$query = "SELECT topic\_id from mismatch\_topic ORDER BY topic\_id";

$result = mysqli\_query($connection, $query) **or** **die**("Collecting Topics Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

array\_push($topicIDs, $row['topic\_id']);

}

*//Insert topic\_id w.r.t user\_id in the mismatch\_response*

**foreach**($topicsIDS **as** $topic\_id){

$query = "INSERT INTO mismatch\_response(user\_id, topic\_id) VALUES('**$cookie\_user\_id**','**$topic\_id**')";

mysqli\_query($connection, $query) **or** **die** ("Inserting Topics Query Denied");

}

**endif**;

*//If the questionnaire form has been submitted, write the form responses to the database*

**if**(isset($\_POST['submit'])):

**foreach**($\_POST **as** $response\_id => $response){

$query = "UPDATE mismatch\_response SET response = '**$response**' WHERE response\_id = '**$response\_id**'";

mysqli\_query($connection, $query) **or** **die** ("Updated the response Query Denied");

}

**endif**;

*/\*\**

*\**

*\* Fetching the response data from the database to generate form*

*\**

*\*/*

*//Collect response\_id, topic\_id, response w.r.t user\_id from mismatch\_response*

$query = "SELECT response\_id, topic\_id, response FROM mismatch\_response WHERE user\_id = '**$cookie\_user\_id**'";

$result = mysqli\_query($connection, $query) **or** **die**("Fetching the data Query Denied");

$responses = **array**();

**while**($response\_row = mysqli\_fetch\_array($result)){

$topic\_id = $response\_row['topic\_id'];

*//Select name, category from the mismatch\_topic using the mismatch\_response topic\_id*

$query2 = "SELECT name, category FROM mismatch\_topic WHERE topic\_id = '**$topic\_id**'";

$topic\_data = mysqli\_query($connection, $query2) **or** ("Fetching category, name Query Denied");

*//Check if there is one topic\_id that matches*

**if**(mysqli\_num\_rows($topic\_data) == 1):

$topic\_row = mysqli\_fetch\_array($topic\_data);

$response\_row['topic\_name'] = $topic\_row['name'];

$response\_row['category\_name'] = $topic\_row['category'];

*//Feeding the topic name, category name to the $responses array*

array\_push($responses, $response\_row);

**endif**;

}

mysqli\_close($connection);

*/\*\**

*\**

*\* Generate the questionnaire form by looping throung the response array*

*\**

*\*/*

**echo** "<form method = 'post' action ='".$\_SERVER['PHP\_SELF']."'>";

**echo** "<p> How do you feel about the each topic? </p>";

*//Display the first category*

$category = $responses[0]['category\_name'];

**echo** "<h3>".$responses[0]['category\_name']."</h3>";

**foreach**($responses **as** $response){

*//Change the category when the category for the response changes*

**if**($category != $response['category\_name']):

$category = $response['category\_name'];

**echo** "<h3>".$response['category\_name']."</h3>";

**endif**;

*//Display the topic form fields*

**echo** "<label for = '".$response['response\_id']."'>".$response['topic\_name']." </label>";

**echo** "<input type = 'radio' name = '".$response['response\_id']."' value = '1'"

.($response['response'] == 1?'checked="checked"' : '')."/>Love";

**echo** "<input type = 'radio' name = '".$response['response\_id']."' value = '2'"

.($response['response'] == 2?'checked="checked"' : '')."/>Hate<br/>";

}

**echo** "<br/><input type = 'submit' value = 'Save' name = 'submit'/>";

**echo** "</form>";

?>

Code 45: question.php

**STEPS**

1. Check whether the mismatch\_response has any topic\_id, response\_id, response w.r.t user\_id in the mismatch\_response
2. If the mismatch\_response has no data, create topic\_id w.r.t user\_id
3. Selecting all the topic\_id from the topic table
4. Insert topic\_id w.r.t user\_id in the mismatch\_response
5. If the questionnaire form has been submitted, write the form responses to the database
6. Collect response\_id, topic\_id, response w.r.t user\_id from mismatch\_response
7. Select name, category from the mismatch\_topic using the mismatch\_response topic\_id
8. Check if there is one topic\_id that matches
9. Feeding the topic name, category name to the $responses array
10. Display the first category
11. Change the category when the category for the response changes
12. Display the topic form fields

#### Normalization

Normalization means designing a database to reduce duplicate data and improve the relationships between data.

Thinking Process:

* The main thing that we want table to be about
* How will we use the table, list of information that we are going to need about that one thing?
* How can we easily query the table?

Atomic data is data that has been broken down into the smallest form needed for a given database. Making our data atomic is the first step in creating a normal table.

Normalization has its benefits, namely improvements in database size and speed. Normal tables won’t have duplicate data which will reduce the size of our database. With less data to search through, our queries will be faster. Normalizing database involves strictly adhering to a series of design steps.

1. Columns should be atomic
2. Each table should have its own primary key
3. Making sure that the non-key columns aren’t dependent on each other.

More tables lead to less messier queries.

A join grabs results from multiple tables in a single query. In an INNER JOIN, it selects two rows from two tables based on a condition.

SELECT mismatch\_topic.topic\_id, mismatch\_category.name

FROM mismatch\_topic

INNER JOIN mismatch\_category

ON (mismatch\_topic.category\_id = mismatch\_category.category\_id)

Dot notation allows us to reference the table a column belongs to within a join.

An INNER JOIN combines rows from two tables using comparison operators in a condition

SELECT mismatch\_topic.topic\_id, mismatch\_category.name

FROM mismatch\_topic

INNER JOIN mismatch\_category

ON (mismatch\_topic.category\_id = mismatch\_category.category\_id)

WHERE mismatch\_topic.name = “Easy Listening Music”

Rewrite on with using for more concise inner join queries that match on a common column. The column names must be the same in order to use the USING statement in an inner join.

SELECT mismatch\_topic.topic\_id, mismatch\_category.name

FROM mismatch\_topic

INNER JOIN mismatch\_category

USING (category\_id)

WHERE mismatch\_topic.name = “Easy Listening Music”

An **Alias** allows us to rename a table or column within a query to help simplify the query in some way. When a column is renamed with an alias the alias is what appears in the query results.

SELECT mt.topic\_id, mc.name

FROM mismatch\_topic AS mt

INNER JOIN mismatch\_category AS mc

USING (category\_id)

WHERE mt.name = “Easy Listening Music”

**Joins** are more efficient and require less code than nested queries. Other types of **Inner Joins** include equijoins, non-equijoins and natural joins. **Equijoins** when the query search for equal columns. **Non-Equijoins** are inequality comparison. **Natural joins** involve comparing all columns that have the same name between two tables. There are several **Outer Joins, Left Outer Join, Right Outer Join, Full Outer Join.** In outer join the rows in the joined table don’t have to match, in order to make it into the join.

**Foreign Key**

CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);

The following SQL creates a **FOREIGN KEY** on the "PersonID" column when the "Orders" table is created:

ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

SQL FOREIGN KEY on ALTER TABLE

ALTER TABLE Orders  
DROP FOREIGN KEY FK\_PersonOrder;

Drop FOREIGN KEY

**Array Slice**

array\_slice(array,start,length,preserve)

<?php  
 $a=array("red","green","blue","yellow","brown");  
 print\_r(array\_slice($a,2));  
?>

Solution:: Array ( [0] => blue [1] => yellow [2] => brown )

Code 46: array\_slice.php

### Search

MySQL searches are by default case-insensitive. SQL queries can be flexible with LIKE

SELECT job\_id title, description FROM jobs WHERE title LIKE ‘%fighter’

The keyword LIKE lets us look for matches that aren’t exactly the same as the word in quotes and still case-insensitive. The % signs are **wildcards,** they stand in any other characters before or after the word. LIKE match the search search as part of a larger word or phrase.

LIKE ‘ \_ \_ \_ \_ fighters%’

Find the string fighter with any four characters in front of it, and any characters after it.

<?php

**require\_once**("common/database.php");

$connection = mysqli\_connect(SERVER, USER, PASS, DB);

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Job Search Engine </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** " <label for = 'search'> Jobs, that you are looking for? </label><br/>";

$search = !**empty**($\_POST['search'])? $\_POST['search'] : '';

**echo** " <input type = 'text' name = 'search' value = '**$search**'/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Search'/>";

**echo** " </form>";

**echo** " </body>";

**echo** " </html>";

**if**(isset($\_POST['submit'])):

**if**(!**empty**($\_POST['search'])){

$search = mysqli\_real\_escape\_string($connection, trim($\_POST['search']));

$query = "SELECT \* FROM riskyjobs WHERE description LIKE '%**$search**%'";

$result = mysqli\_query($connection, $query) **or** **die** ("Search Query Denied");

**echo** "<table>";

**echo** "<tr>";

**echo** " <th>Title</th>";

**echo** " <th>Description</th>";

**echo** " <th>City</th>";

**echo** " <th>State</th>";

**echo** " <th>Zip</th>";

**echo** " <th>Company</th>";

**echo** " <th>Date Posted</th>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** " <td>".$row['title']."</td>";

**echo** " <td>".$row['description']."</td>";

**echo** " <td>".$row['city']."</td>";

**echo** " <td>".$row['state']."</td>";

**echo** " <td>".$row['zip']."</td>";

**echo** " <td>".$row['company']."</td>";

**echo** " <td>".$row['date\_posted']."</td>";

**echo** "</tr>";

}

**echo** "</table>";

}

**else**{

$error = "You cannot leave the search box empty!";

**echo** "<span>".$error."</span>";

}

**endif**;

?>

Code 46: search.php

#### Break Up the Phrases

The **explode()** function breaks a string into an array of substrings.

Ex: $search\_words = explode(‘ ‘, ‘Hulk Hogan’);

The $search\_words variable now stores the array of search terms. ‘ ‘ 🡪 This is what separated the substrings within the string, in this case a space. We can specify one or more characters, which are called **delimiter**.

The **implode()** function takes an array of strings and builds a single string out of them.

Ex: $string\_attach = implode(‘-‘, $list);

$together = ‘’;

$lists = [‘apple’, ‘orange’, ‘banana’];

foreach($lists as $item){

$together .= $item;

}

echo($together);

**Solution:** appleorangebanana

Code 47: stringattach.php

<?php

**require\_once**("common/database.php");

$connection = mysqli\_connect(SERVER, USER, PASS, DB);

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Job Search Engine </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** " <label for = 'search'> Jobs, that you are looking for? </label><br/>";

$search = !**empty**($\_POST['search'])? $\_POST['search'] : '';

**echo** " <input type = 'text' name = 'search' value = '**$search**'/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Search'/>";

**echo** " </form>";

**echo** " </body>";

**echo** " </html>";

**if**(isset($\_POST['submit'])):

**if**(!**empty**($\_POST['search'])){

$search = mysqli\_real\_escape\_string($connection, trim($\_POST['search']));

$search\_words = explode(' ', $search);

**echo** "<table>";

**echo** "<tr>";

**echo** " <th>Title</th>";

**echo** " <th>Description</th>";

**echo** " <th>City</th>";

**echo** " <th>State</th>";

**echo** " <th>Zip</th>";

**echo** " <th>Company</th>";

**echo** " <th>Date Posted</th>";

**echo** "</tr>";

*// foreach($search\_words as $search){*

*// $query = "SELECT \* FROM riskyjobs WHERE description LIKE '%$search%'";*

*// $result = mysqli\_query($connection, $query) or die ("Search Query Denied");*

*// while($row = mysqli\_fetch\_array($result)){*

*// echo "<tr>";*

*// echo " <td>".$row['title']."</td>";*

*// echo " <td>".$row['description']."</td>";*

*// echo " <td>".$row['city']."</td>";*

*// echo " <td>".$row['state']."</td>";*

*// echo " <td>".$row['zip']."</td>";*

*// echo " <td>".$row['company']."</td>";*

*// echo " <td>".$row['date\_posted']."</td>";*

*// echo "</tr>";*

*// }*

*// }*

*/\*\**

*\* Alternatively*

*\*/*

$where\_list = **array**();

**foreach**($search\_words **as** $word){

$where\_list[] = " description LIKE '%**$word**%' ";

}

$where\_clause = implode('OR', $where\_list);

**if**(!**empty**($where\_clause)):

$where\_clause = "WHERE **$where\_clause**";

**endif**;

$query = "SELECT \* FROM riskyjobs **$where\_clause**";

$result = mysqli\_query($connection, $query) **or** **die** ("Search Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** " <td>".$row['title']."</td>";

**echo** " <td>".$row['description']."</td>";

**echo** " <td>".$row['city']."</td>";

**echo** " <td>".$row['state']."</td>";

**echo** " <td>".$row['zip']."</td>";

**echo** " <td>".$row['company']."</td>";

**echo** " <td>".$row['date\_posted']."</td>";

**echo** "</tr>";

}

**echo** "</table>";

}

**else**{

$error = "You cannot leave the search box empty!";

**echo** "<span>".$error."</span>";

}

**endif**;

?>

Code 47: brokenphrase.php

**Preprocessing data** allows us to remove unwanted characters and make the data easier to process.

str\_replace(‘delimiter’, ‘replace\_item’, $string);

The substring ‘delimiter’ gets replaced by ‘replace item’ in the string

<?php

**require\_once**("common/database.php");

$connection = mysqli\_connect(SERVER, USER, PASS, DB);

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Job Search Engine </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** " <label for = 'search'> Jobs, that you are looking for? </label><br/>";

$search = !**empty**($\_POST['search'])? $\_POST['search'] : '';

**echo** " <input type = 'text' name = 'search' value = '**$search**'/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Search'/>";

**echo** " </form>";

**echo** " </body>";

**echo** " </html>";

**if**(isset($\_POST['submit'])):

**if**(!**empty**($\_POST['search'])){

$search = mysqli\_real\_escape\_string($connection, trim($\_POST['search']));

$clean\_words = str\_replace(","," ",$search);

$search\_words = explode(' ', $clean\_words);

$final\_words = **array**();

**foreach**($search\_words **as** $words){

**if**(!**empty**($words)):

$final\_words[] = $words;

**endif**;

}

**echo** "<table>";

**echo** "<tr>";

**echo** " <th>Title</th>";

**echo** " <th>Description</th>";

**echo** " <th>City</th>";

**echo** " <th>State</th>";

**echo** " <th>Zip</th>";

**echo** " <th>Company</th>";

**echo** " <th>Date Posted</th>";

**echo** "</tr>";

$where\_list = **array**();

**foreach**($final\_words **as** $word){

$where\_list[] = " description LIKE '%**$word**%' ";

}

$where\_clause = implode('OR', $where\_list);

**if**(!**empty**($where\_clause)):

$where\_clause = "WHERE **$where\_clause**";

**endif**;

$query = "SELECT \* FROM riskyjobs **$where\_clause**";

$result = mysqli\_query($connection, $query) **or** **die** ("Search Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** " <td>".$row['title']."</td>";

**echo** " <td>".$row['description']."</td>";

**echo** " <td>".$row['city']."</td>";

**echo** " <td>".$row['state']."</td>";

**echo** " <td>".$row['zip']."</td>";

**echo** " <td>".$row['company']."</td>";

**echo** " <td>".$row['date\_posted']."</td>";

**echo** "</tr>";

}

**echo** "</table>";

}

**else**{

$error = "You cannot leave the search box empty!";

**echo** "<span>".$error."</span>";

}

**endif**;

?>

Code 48: replacestring.php

substr(string, start, length)

The PHP function **substr()** function allows us to extract a portion of a string**.**

$string = “Hello world”

0 1 2 3 4 5 6 7 8 9

-10-9-8-7-6 -5-4 -3 -2 -1

substr($row['description'], 0,100)

substr($row['date\_posted'], 0, 9)

#### Custom Functions

**Customs functions** allow us to organize a chunk of PHP code by name so that it can be easily reused.

A function is a block of code separate from the rest of our code that can execute where we want in our script.

<?php

**require\_once**("common/database.php");

**require\_once**("buildquery.php");

**require\_once**("sort.php");

**require\_once**("generate\_sort\_links.php");

$connection = mysqli\_connect(SERVER, USER, PASS, DB);

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Job Search Engine </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <form method = 'get' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** " <label for = 'search'> Jobs, that you are looking for? </label><br/>";

$search = !**empty**($\_GET['search'])? $\_GET['search'] : '';

**echo** " <input type = 'text' name = 'search' value = '**$search**'/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Search'/>";

**echo** " </form><br/>";

**echo** " </body>";

**echo** " </html>";

**if**(isset($\_GET['submit'])):

**if**(!**empty**($\_GET['search'])){

$sortby = (!**empty**($\_GET['sort'])) ? $\_GET['sort'] : 1;

**echo** generate\_sort\_links($\_GET['search'], $sortby);

$search = mysqli\_real\_escape\_string($connection, trim($\_GET['search']));

$where\_clause = build\_query($search);

$order\_by = sorting($sortby);

$query = "SELECT \* FROM riskyjobs **$where\_clause** **$order\_by**";

$result = mysqli\_query($connection, $query) **or** **die** ("Search Query Denied");

**echo** "<table>";

**echo** "<tr>";

**echo** " <th>Title</th>";

**echo** " <th>Description</th>";

**echo** " <th>City</th>";

**echo** " <th>State</th>";

**echo** " <th>Zip</th>";

**echo** " <th>Company</th>";

**echo** " <th>Date Posted</th>";

**echo** "</tr>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** " <td>".$row['title']."</td>";

**echo** " <td>".substr($row['description'], 0,100)."</td>";

**echo** " <td>".$row['city']."</td>";

**echo** " <td>".$row['state']."</td>";

**echo** " <td>".$row['zip']."</td>";

**echo** " <td>".$row['company']."</td>";

**echo** " <td>".substr($row['date\_posted'], 0, 9)."</td>";

**echo** "</tr>";

}

**echo** "</table>";

}

**else**{

$error = "You cannot leave the search box empty!";

**echo** "<span>".$error."</span>";

}

**endif**;

?>

Code 49: replacestring.php

<?php

**function** build\_query($post\_search){

$clean\_words = str\_replace(","," ",$post\_search);

$search\_words = explode(' ', $clean\_words);

$final\_words = **array**();

**foreach**($search\_words **as** $words){

**if**(!**empty**($words)):

$final\_words[] = $words;

**endif**;

}

$where\_list = **array**();

**foreach**($final\_words **as** $word){

$where\_list[] = " description LIKE '%**$word**%' ";

}

$where\_clause = implode('OR', $where\_list);

**if**(!**empty**($where\_clause)):

$where\_clause = "WHERE **$where\_clause**";

**endif**;

**return** $where\_clause;

}

?>

Code 50: buildquery.php

<?php

**function** sorting($sortby){

$string = 'ORDER BY ';

**switch**($sortby){

**case** 1:

$string .= " title ASC";

**break**;

**case** 2:

$string .= " title DESC";

**break**;

**case** 3:

$string .= " state ASC";

**break**;

**case** 4:

$string .= " state DESC";

**break**;

**case** 5:

$string .= " date\_posted ASC";

**break**;

**case** 6:

$string .= " date\_posted DESC";

**break**;

**default**:

}

**return** $string;

}

?>

Code 51: sort.php

<?php

**function** generate\_sort\_links($user\_search, $sort){

$string = '';

$self = $\_SERVER['PHP\_SELF'];

**switch**($sort){

**case** 1:

$string .= "<p><a href = '".$self."?search=".$user\_search."&amp;sort=2&amp;submit=1'>Job Title</a> /";

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=3&amp;submit=1'>State</a> /";

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=5&amp;submit=1'>Date Posted</a></p>";

**break**;

**case** 3:

$string .= "<p><a href = '".$self."?search=".$user\_search."&amp;sort=1&amp;submit=1'>Job Title</a> / ";

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=4&amp;submit=1'>State</a> / ";

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=5&amp;submit=1'>Date Posted</a></p>";

**break**;

**case** 5:

$string .= "<p><a href = '".$self."?search=".$user\_search."&amp;sort=1&amp;submit=1'>Job Title</a> / ";

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=3&amp;submit=1'>State</a> / ";

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=6&amp;submit=1'>Date Posted</a></p>";

**break**;

**default**:

$string .= "<p><a href = '".$self."?search=".$user\_search."&amp;sort=1&amp;submit=1'>Job Title</a> / ";

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=3&amp;submit=1'>State</a> / ";

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=5&amp;submit=1'>Date Posted</a></p>";

}

**return** $string;

}

?>

Code 52: generate\_sort\_links.php

#### Pagination

**Pagination** breaks query results into sets, and display search set on its own web page. Limit controls what and how many rows are returned by an SQL query. When two arguments are given to the Limit SQL, the first arguments controls how many rows we skip, and the second argument controls how many rows we get back.

SELECT FROM table\_name WHERE COLUMN = match LIMIT num(skip) num(show)

<?php

**function** generate\_page\_links($user\_search, $sortby, $cur\_page, $num\_pages){

$string = '';

$self = $\_SERVER['PHP\_SELF'];

$previous = $cur\_page - 1;

$next = $cur\_page + 1;

*//If this is not the first page, generate the "Previous Links"*

**if**($cur\_page > 1){

$string .= "<p><a href = '".$self."?search=".$user\_search."&amp;sort=**$sortby**&amp;submit=1"

."&amp;page=**$previous**'><--</a>";

}

**else**{

$string .=' ';

}

**for**($i = 1; $i <= $num\_pages; $i++){

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=**$sortby**&amp;submit=1"

."&amp;page=**$i**'>**$i**</a>";

}

**if**($cur\_page < $num\_pages){

$string .= "<a href = '".$self."?search=".$user\_search."&amp;sort=**$sortby**&amp;submit=1"

."&amp;page=**$next**'>--></a></p>";

}

**else**{

$string .=' ';

}

**return** $string;

}

?>

Code 52: generate\_pagination\_links.php

<?php

**require\_once**("common/database.php");

**require\_once**("buildquery.php");

**require\_once**("sort.php");

**require\_once**("generate\_sort\_links.php");

**require\_once**("generate\_page\_links.php");

$connection = mysqli\_connect(SERVER, USER, PASS, DB);

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Job Search Engine </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <form method = 'get' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** " <label for = 'search'> Jobs, that you are looking for? </label><br/>";

$search = !**empty**($\_GET['search'])? $\_GET['search'] : '';

**echo** " <input type = 'text' name = 'search' value = '**$search**'/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Search'/>";

**echo** " </form><br/>";

**echo** " </body>";

**echo** " </html>";

**if**(isset($\_GET['submit'])):

**if**(!**empty**($\_GET['search'])){

*//Sort value from GET*

$sortby = (!**empty**($\_GET['sort'])) ? $\_GET['sort'] : 1;

*//Generating the links to order the query*

**echo** generate\_sort\_links($\_GET['search'], $sortby);

*//Get the search data*

$search = mysqli\_real\_escape\_string($connection, trim($\_GET['search']));

*//Clean up the data*

$where\_clause = build\_query($search);

*//Sort the data*

$order\_by = sorting($sortby);

*//Find out the number of rows*

$query = "SELECT \* FROM riskyjobs **$where\_clause** **$order\_by**";

$result = mysqli\_query($connection, $query) **or** **die** ("Total Query Denied");

*//Calculating the LIMIT skip for the query*

$cur\_page = isset($\_GET['page'])? $\_GET['page']: 1;

$results\_per\_page = 5;

$skip = ($cur\_page - 1) \* $results\_per\_page;

$total\_query = mysqli\_num\_rows($result);

$num\_pages = ceil($total\_query/$results\_per\_page);

*//Generate pagination link*

**echo** generate\_page\_links($\_GET['search'], $sortby, $cur\_page, $num\_pages);

$query = "SELECT \* FROM riskyjobs **$where\_clause** **$order\_by** LIMIT **$skip**, **$results\_per\_page**";

**echo** $query;

$result = mysqli\_query($connection, $query) **or** **die** ("Search Query Denied");

**echo** "<table>";

**echo** "<tr>";

**echo** " <th>Title</th>";

**echo** " <th>Description</th>";

**echo** " <th>City</th>";

**echo** " <th>State</th>";

**echo** " <th>Zip</th>";

**echo** " <th>Company</th>";

**echo** " <th>Date Posted</th>";

**echo** "</tr>";

**while**($row = mysqli\_fetch\_array($result)){

**echo** "<tr>";

**echo** " <td>".$row['title']."</td>";

**echo** " <td>".substr($row['description'], 0,100)."</td>";

**echo** " <td>".$row['city']."</td>";

**echo** " <td>".$row['state']."</td>";

**echo** " <td>".$row['zip']."</td>";

**echo** " <td>".$row['company']."</td>";

**echo** " <td>".substr($row['date\_posted'], 0, 9)."</td>";

**echo** "</tr>";

}

**echo** "</table>";

}

**else**{

$error = "You cannot leave the search box empty!";

**echo** "<span>".$error."</span>";

}

**endif**;

?>

Code 52: replacestring.php

ceil($total\_query/$results\_per\_page)

This rounds up the value

### Regular Expression

#### Validation

<?php

$show\_form = **TRUE**;

$error = **array**();

**if**(isset($\_POST['submit'])):

$first\_name = $\_POST['first\_name'];

$last\_name = $\_POST['last\_name'];

$email = $\_POST['email'];

$phone = $\_POST['phone'];

$desired\_job = $\_POST['desired\_job'];

$resume\_paste = $\_POST['resume\_paste'];

**if**(**empty**($first\_name)):

$error['first\_name'] = "Please enter your first name!";

$show\_form = **TRUE**;

**endif**;

**if**(**empty**($last\_name)):

$error['last\_name'] = "Please enter your last name!";

$show\_form = **TRUE**;

**endif**;

**if**(**empty**($email)):

$error['email'] = "Please enter your email!";

$show\_form = **TRUE**;

**endif**;

**if**(**empty**($phone)):

$error['phone'] = "Please enter your phone number!";

$show\_form = **TRUE**;

**endif**;

**if**(**empty**($desired\_job)):

$error['desired\_job'] = "Please fill in your desired job!";

$show\_form = **TRUE**;

**endif**;

**if**(**empty**($resume\_paste)):

$error['resume\_paste'] = "Please fill in your resume!";

$show\_form = **TRUE**;

**endif**;

**endif**;

**if**($show\_form):

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Resume </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** " <label for = 'first\_name'> First Name </label><br/>";

**echo** " <input type = 'text' name = 'first\_name'/>";

$first\_error = (!**empty**($error['first\_name']))? $error['first\_name'] : "";

**echo** " <span>".$first\_error."</span>";

**echo** " <br/>";

**echo** " <label for = 'last\_name'> Last Name </label><br/>";

**echo** " <input type = 'text' name = 'last\_name'/>";

$last\_error = (!**empty**($error['last\_name']))? $error['last\_name'] : "";

**echo** " <span>".$last\_error."</span>";

**echo** " <br/>";

**echo** " <label for = 'email'> Email </label><br/>";

**echo** " <input type = 'email' name = 'email'/>";

$email\_error = (!**empty**($error['email']))? $error['email'] : "";

**echo** " <span>".$email\_error."</span>";

**echo** " <br/>";

**echo** " <label for = 'phone'> Phone </label><br/>";

**echo** " <input type = 'text' name = 'phone'>";

$phone\_error = (!**empty**($error['phone']))? $error['phone'] : "";

**echo** " <span>".$phone\_error."</span>";

**echo** " <br/>";

**echo** " <label for = 'desired\_job'> Desired Job </label><br/>";

**echo** " <input type = 'text' name = 'desired\_job'/>";

$desired\_error = (!**empty**($error['desired\_job']))? $error['desired\_job'] : "";

**echo** " <span>".$desired\_error."</span>";

**echo** " <br/>";

**echo** " <label for = 'resume\_paste'> Resume Paste </label><br/>";

**echo** " <textarea name = 'resume\_paste'></textarea>";

$resume\_error = (!**empty**($error['resume\_paste']))? $error['resume\_paste'] : "";

**echo** " <span>".$resume\_error."</span>";

**echo** " <br/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Confirm'/>";

**echo** " </form>";

**echo** " </body>";

**echo** " </html>";

**endif**;

?>

Code 53: resume.php

**Regular Expressions** are rules used to match patterns in one or more strings.

Example: US Phone numbers has 0 digits

**/^\d\d\d\d\d\d\d\d\d\d$/**

**^** **Carat,** start matching at the begging of the string

**\d Digit,** the first character in the string must be a digit.

**$ Dollar,** string must end

**| Vertical Pipe,** indicates sets of options to choose from

Now this can be replaced with **/^{d10}$/**

**{10},** shorthand way to say 10 digits

**Metacharacters** let us describe patterns of text within a regular expression

**\d,** any number from **0 to 9**

**\w,** any alphanumeric character, either a letter or number **a-z, A-Z, 0-9**

**\s,** whitespace, tab character, or a newline or carriage return

**^,** looking for the start of the string “**40 soldiers**” from “**We have 40 soldiers**”

**$,** looking for the end of the string

**.,** matches any one character except newline. It will match **letter, digit, space or tab**

Regular Expressions(Regex) supports a feature called quantifier. **A Quantifier** specifies how many times a metacharacter should appear.

**{min,max}** A metacharacter should appear between min to max times

**+** A metacharacter must appear **one or more times**

**\*** The metacharacter must appear **one or more times or not at all**

**?** The metacharacter must appear **once or not at all**

Therefore, US 9-digit number with 4-digit extension(optional) would be

**/^\d{3}-\d{3}-d{3}(-\d{4})?$/**

A **character class** is a set of rules for matching a single character.

**[0-2],** this matches a range of numbers. It will match **0, 1, 2**

**[A-D],** This will match **A, B, C and D**

**[^b-f],** Match everything **except b to f**

In united states the first digit 0 connects to an operator, and 1 dials long distance

Therefore, **/^[0-2]\d{2}-\d{3}-\d{3}(-\d{4})?$/**

[m-z, M-Z], those extra spaces will be interpreted as part of the set of characters that should match the text string

We can use reserve characters in our regular expressions by escaping them.

preg\_match($regex, $my\_string)

This function takes a regex pattern and a text string, it returns true if it matches

preg\_replace($pattern, $replacement, $my\_string)

The pattern gets replaced by ‘replacement’ in the string

Standardizing our data gives us better SQL query results.

**(###)-###-#### 🡪 ##########**

To remove it we can use preg\_replace($regex, $replacement, $string)

$regex = “/^\(\)\-\s$/”

$replacement = “”

Validation is often a trade-off between what’s allowed and what is practical to accept.

**Domain name system** is a distributed data service provides a worldwide directory of domains and their IP addresses. It names possible. Without DNS, we would be typing 208.201.239.36. We can check the domain portion of the email address using the PHP function **checkdnsrr($domain)**

For email, $regex = "/^[a-zA-Z0-9\\_\.][+@[a-zA-Z0-9]+\.\w{3}$/](mailto:+@[a-zA-Z0-9]+\.\w%7b3%7d$/)";

<?php

$show\_form = **TRUE**;

$error = **array**();

**if**(isset($\_POST['submit'])){

$first\_name = $\_POST['first\_name'];

$last\_name = $\_POST['last\_name'];

$email = $\_POST['email'];

$phone = $\_POST['phone'];

$desired\_job = $\_POST['desired\_job'];

$resume\_paste = $\_POST['resume\_paste'];

**if**(**empty**($first\_name)){

$error['first\_name'] = "Please enter your first name!";

$show\_form = **TRUE**;

}

**if**(**empty**($last\_name)){

$error['last\_name'] = "Please enter your last name!";

$show\_form = **TRUE**;

}

**if**(**empty**($email)){

$error['email'] = "Please enter your email!";

$show\_form = **TRUE**;

}

**if**(!**empty**($email)){

*// $regex = "/^[a-zA-Z0-9\\_\.]+@[a-zA-Z0-9]+\.\w{3}$/";*

$regex = "/^[a-zA-Z0-9\\_\.]+@/";

**if**(!preg\_match($regex, $email)){

$error['email\_format'] = "xxxx@xxxx.xxx format do not match!";

}

**else**{

$domain = preg\_replace($regex, '', $email);

**if**(!checkdnsrr($domain)){

$error['email\_format'] = "Domain do not match!";

}

}

}

**if**(**empty**($phone)){

$error['phone'] = "Please enter your phone number!";

$show\_form = **TRUE**;

}

**if**(!**empty**($phone)){

$regex = "/^[0-2]\d{2}-\d{3}-\d{3}(\d{4})?$/";

**if**(!preg\_match($regex, $phone)){

$error['phone\_format'] = "XXX-XXX-XXX format do not match!";

}

}

**if**(**empty**($desired\_job)){

$error['desired\_job'] = "Please fill in your desired job!";

$show\_form = **TRUE**;

}

**if**(**empty**($resume\_paste)){

$error['resume\_paste'] = "Please fill in your resume!";

$show\_form = **TRUE**;

}

}

**if**($show\_form){

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title> Resume </title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** " <label for = 'first\_name'> First Name </label><br/>";

$first\_name =(!**empty**($\_POST['first\_name']))? $\_POST['first\_name'] : "";

**echo** " <input type = 'text' name = 'first\_name' value = '"

.$first\_name

."'/>";

$first\_error = (!**empty**($error['first\_name']))? $error['first\_name'] : "";

**echo** " <span>".$first\_error."</span>";

**echo** " <br/>";

**echo** " <label for = 'last\_name'> Last Name </label><br/>";

$last\_name =(!**empty**($\_POST['last\_name']))? $\_POST['last\_name'] : "";

**echo** " <input type = 'text' name = 'last\_name' value ='"

.$last\_name

."'/>";

$last\_error = (!**empty**($error['last\_name']))? $error['last\_name'] : "";

**echo** " <span>".$last\_error."</span>";

**echo** " <br/>";

**echo** " <label for = 'email'> Email </label><br/>";

$email =(!**empty**($\_POST['email']))? $\_POST['email'] : "";

**echo** " <input type = 'text' name = 'email' value = '"

.$email

."'/>";

$email\_error = (!**empty**($error['email']))? $error['email'] : "";

$email\_format = (!**empty**($error['email\_format']))? $error['email\_format'] : "";

**echo** " <span>".$email\_error." ".$email\_format."</span>";

**echo** " <br/>";

**echo** " <label for = 'phone'> Phone </label><br/>";

$phone =(!**empty**($\_POST['phone']))? $\_POST['phone'] : "";

**echo** " <input type = 'text' name = 'phone' value = '"

.$phone

."'/>";

$phone\_error = (!**empty**($error['phone']))? $error['phone'] : "";

$phone\_format = (!**empty**($error['phone\_format']))? $error['phone\_format'] : "";

**echo** " <span>".$phone\_error." ".$phone\_format."</span>";

**echo** " <br/>";

**echo** " <label for = 'desired\_job'> Desired Job </label><br/>";

$desired\_job =(!**empty**($\_POST['desired\_job']))? $\_POST['desired\_job'] : "";

**echo** " <input type = 'text' name = 'desired\_job' value ='"

.$desired\_job

."'/>";

$desired\_error = (!**empty**($error['desired\_job']))? $error['desired\_job'] : "";

**echo** " <span>".$desired\_error."</span>";

**echo** " <br/>";

**echo** " <label for = 'resume\_paste'> Resume Paste </label><br/>";

$resume\_paste = (!**empty**($\_POST['resume\_paste']))? $\_POST['resume\_paste'] : "";

**echo** " <textarea name = 'resume\_paste'>"

.$resume\_paste

."</textarea>";

$resume\_error = (!**empty**($error['resume\_paste']))? $error['resume\_paste'] : "";

**echo** " <span>".$resume\_error."</span>";

**echo** " <br/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Confirm'/>";

**echo** " </form>";

**echo** " </body>";

**echo** " </html>";

}

?>

Code 54: resumevalidation.php

### Graphics

All web forms are at risk of attack from spam bots.

A **CAPTCHA** is a program that protects a web site from automated bots by using a test of some sort. **Completely Automated Public Turing Test to Tell Computers and Humans Apart**. It is optical character recognition test for humans.

PHP has graphics capabilities that can dynamically generate images that can then display using HTML code. Its graphic library is called **GD (Graphics Draw).**

rand()

This function returns a random integer within a certain range. This built in function returns a random integer number either **within a specified range** or **between 0 and** the built-in constant **RAND\_MAX** (server dependent). To obtain a random number within a certain range, just pass the lower and upper limits of the range as two arguments to rand()

chr()

This build-in function converts a number to its ASCII character equivalent. As an example, the number 97 is the ASCII code for the lowercase let ‘a’. So calling chr(97) return the single character ‘a’

<?php

$no\_of\_words = 5;

$width = 100;

$height = 20;

$phrase = "";

**for**($i = 0; $i < $no\_of\_words; $i++){

$phrase .= chr(rand(97,122));

}

$image = imagecreatetruecolor($width, $height);

$bgd\_color = imagecolorallocate($image, 255, 255, 255);

$text\_color = imagecolorallocate($image, 255, 0 , 0);

$graphic\_color = imagecolorallocate($image, 0, 255, 0);

$graphic\_color2 = imagecolorallocate($image, 0, 0, 255);

imagefilledrectangle($image, 0, 0, $width, $height, $bgd\_color);

**for**($i = 1; $i < $no\_of\_words; $i++){

imageline($image, 0, rand(0, $height), rand(0, $width), rand(0, $height), $graphic\_color);

}

**for**($i = 1; $i < ($no\_of\_words \* 10); $i++){

imagesetpixel($image, rand(0, $width), rand(0, $height), $graphic\_color2);

}

imagestring($image, 5, 25, 3, $phrase, $text\_color);

header("Cache-Control: no-cache, must-revalidate");

header("Content-Type: image/png");

imagepng($image);

imagedestroy($image);

?>

Code 55: captcha.php

$image = **imagecreatetruecolor($width, $height)**

The function creates a blank image in memory ready to be drawn to with other GD functions. This is an **image identifier**. Then to change the background color before drawing anything we need to call



Fig 19: imagecreatetruecolor

**imagefilledrectangle($image, x1, y1, x2, y2, $color)**

It draws a rectangle whose interior is filled with the specified color.



Fig 20: imagefilledrectangle

**imagecolorallocate($image, red, green, blue)**

It is **color identifier**

****

Fig 21: imageecolorallocate

**imagesetpixel($image, rand(0, $width), rand(0, $height), $graphic\_color)**

The function draws a single pixel at a specified coordinate within the image. Coordinates start at 0,0 in the upper left corner of the image and increase to the right and down.



Fig 22: imagesetpixel

**imageline($image, 0, rand(0, $height), rand(0, $width), rand(0, $height), $graphic\_color)**

This function draws a line between two co-ordinates (x1, y1 and x2, y2)



Fig 23: imageline

**imagefilledellipse(image, cx, cy, width, height, color)**

For drawing circles and ellipses, this function accepts a center point and a width and height. A perfect circle is just an ellipse with an equal height and width



Fig 24: imagefilledellipse

**imagefilledarc(image, cx, cy, width, height, start, end, color, style)**

**imagefilledpolygon(image, points, num\_points, color)**

These function works similar like the ellipses one

**imagepng(image, filename, 5)**

This function returns true or false depending on whether the image was successfully created.

We must call the header the function to have it delivered to the browser via header. Writes the image to a PNG file with the specified filename and a compression level of 5

**imagedestroy($image)**

Always free up images in memory with imagedestroy() one we have output them

**imagestring($image, 5, 25, 3, $phrase, $text\_color);**

This function draws a string of text using PHP’s built-in font in the color specified.

Screen Clipping

Fig 25: imagestring

**imagestringup(image, font, x, y, string, color)**

This function draw the text vertically.

Screen Clipping

Fig 26: imagestringup

**imagettftext(image, size, angle, x, y, color, fontfile, text)**

Use the imagettftext() function to draw highly customized text with our own TrueType font.

**imagegif(image)**

It is used to create gif image

**imagejpeg(image)**

It is used to create jpeg image

**imagecolortranparent($image)**

This sets color as a transparent color with an image. This must be a color that we have created using imagecolorallocate(). To generate the image, just either imagegif() or imagepng(). image(jpeg) doesn’t support transparency

**sha1(str) or md5(str)**

PHP offers encryption

<?php

$no\_of\_words = 5;

$width = 100;

$height = 20;

$phrase = "";

**for**($i = 0; $i < $no\_of\_words; $i++){

$phrase .= chr(rand(97,122));

}

$image = imagecreatetruecolor($width, $height);

$bgd\_color = imagecolorallocate($image, 255, 255, 255);

$text\_color = imagecolorallocate($image, 255, 0 , 0);

$graphic\_color = imagecolorallocate($image, 0, 255, 0);

$graphic\_color2 = imagecolorallocate($image, 0, 0, 255);

imagefilledrectangle($image, 0, 0, $width, $height, $bgd\_color);

**for**($i = 1; $i < $no\_of\_words; $i++){

imageline($image, 0, rand(0, $height), rand(0, $width), rand(0, $height), $graphic\_color);

}

**for**($i = 1; $i < ($no\_of\_words \* 10); $i++){

imagesetpixel($image, rand(0, $width), rand(0, $height), $graphic\_color2);

}

imagettftext($image, 15, 0, 22, 18, $text\_color, "fonts/ubuntu/Ubuntu-regular.ttf", $phrase);

header("Cache-Control: no-cache, must-revalidate");

header("Content-Type: image/png");

imagepng($image);

imagedestroy($image);

?>

Code 55: captcha2.php

<?php

session\_start();

$no\_of\_words = 5;

$width = 100;

$height = 20;

$phrase = "";

**for**($i = 0; $i < $no\_of\_words; $i++){

$phrase .= chr(rand(97,122));

}

$\_SESSION['phrase'] = sha1($phrase);

setcookie('phrase', sha1($phrase), time() + (3600 \* 30), "/");

$image = imagecreatetruecolor($width, $height);

$bgd\_color = imagecolorallocate($image, 255, 255, 255);

$text\_color = imagecolorallocate($image, 255, 0 , 0);

$graphic\_color = imagecolorallocate($image, 0, 255, 0);

$graphic\_color2 = imagecolorallocate($image, 0, 0, 255);

imagefilledrectangle($image, 0, 0, $width, $height, $bgd\_color);

**for**($i = 1; $i < $no\_of\_words; $i++){

imageline($image, 0, rand(0, $height), rand(0, $width), rand(0, $height), $graphic\_color);

}

**for**($i = 1; $i < ($no\_of\_words \* 10); $i++){

imagesetpixel($image, rand(0, $width), rand(0, $height), $graphic\_color2);

}

imagettftext($image, 15, 0, 22, 18, $text\_color, "fonts/ubuntu/Ubuntu-regular.ttf", $phrase);

header("Cache-Control: no-cache, must-revalidate");

header("Content-Type: image/png");

imagepng($image);

imagedestroy($image);

?>

Code 56: captcha.php

<?php

**require\_once**("common/database.php");

**require\_once**("common/constants.php");

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die** ("Server Denied");

$showform = **TRUE**;

**if**(isset($\_POST['submit'])):

**if**(

!**empty**($\_POST['username']) &&

!**empty**($\_POST['password']) &&

!**empty**($\_POST['first\_name']) &&

!**empty**($\_POST['last\_name']) &&

!**empty**($\_POST['gender']) &&

!**empty**($\_POST['date']) &&

!**empty**($\_POST['city']) &&

!**empty**($\_POST['state']) &&

!**empty**($\_FILES['picture']) &&

$\_POST['password'] == $\_POST['password\_two'] &&

$\_COOKIE['phrase'] == sha1($\_POST['captcha'])

):

$showform = **FALSE**;

$username = isset($\_POST['username'])? mysqli\_real\_escape\_string($connection, trim($\_POST['username'])): **NULL**;

$password = isset($\_POST['password'])? mysqli\_real\_escape\_string($connection, trim($\_POST['password'])): **NULL**;

$first\_name = isset($\_POST['first\_name'])? mysqli\_real\_escape\_string($connection, trim($\_POST['first\_name'])): **NULL**;

$last\_name = isset($\_POST['last\_name'])? mysqli\_real\_escape\_string($connection, trim($\_POST['last\_name'])): **NULL**;

$gender = isset($\_POST['gender'])? mysqli\_real\_escape\_string($connection, trim($\_POST['gender'])): **NULL**;

$date = isset($\_POST['date'])? mysqli\_real\_escape\_string($connection, trim($\_POST['date'])): **NULL**;

$city = isset($\_POST['city'])? mysqli\_real\_escape\_string($connection, trim($\_POST['city'])): **NULL**;

$state = isset($\_POST['state'])? mysqli\_real\_escape\_string($connection, trim($\_POST['state'])): **NULL**;

$picture = isset($\_FILES['picture'])? mysqli\_real\_escape\_string($connection, trim($\_FILES['picture']['name'])): **NULL**;

$query = "INSERT INTO mismatch\_user(username, password, join\_date, first\_name, last\_name, gender, birth\_date, city, state, picture)

VALUES('**$username**', SHA('**$password**'), NOW(), '**$first\_name**', '**$last\_name**', '**$gender**', '**$date**', '**$city**', '**$state**', '**$picture**')";

mysqli\_query($connection, $query);

$destination = UPLOAD\_PATH.$\_FILES['picture']['name'];

move\_uploaded\_file($\_FILES['picture']['tmp\_name'], $destination);

**if**(isset($\_SESSION['phrase']) && !**empty**($\_SESSION['phrase'])):

$\_SESSION = **array**();

**if**(isset($\_COOKIE[session\_name()])):

setcookie(session\_name(), '', time() - 3600, '/');

**endif**;

session\_destroy();

**endif**;

**if**(isset($\_COOKIE['phrase']) && !**empty**($\_COOKIE['phrase'])):

setcookie('phrase',' ', time() - 3600, "/");

**endif**;

**echo** "User Successfully Added";

**endif**;

**endif**;

**if**($showform):

**echo** '<!doctype html>';

**echo** ' <html>';

**echo** ' <head>';

**echo** ' <title> Add Users </title>';

**echo** ' <meta charset = "UTF-8"/>';

**echo** ' </head>';

**echo** ' <body>';

**echo** " <form enctype = 'multipart/form-data' method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** ' <label for = "username">Username</label><br/>';

**echo** ' <input type = "text" name = "username"/><br/>';

**echo** ' <label for = "password">Password</label><br/>';

**echo** ' <input type = "password" name = "password"/><br/>';

**echo** ' <label for = "password\_two">Retype Password</label><br/>';

**echo** ' <input type = "password" name = "password\_two"/><br/>';

**echo** ' <label for = "first\_name">First Name</label><br/>';

**echo** ' <input type = "text" name = "first\_name"/><br/>';

**echo** ' <label for = "last\_name">Last Name</label><br/>';

**echo** ' <input type = "text" name = "last\_name"/><br/>';

**echo** ' <label for = "gender">Gender</label><br/>';

**echo** ' Male<input type = "radio" name = "gender" value = "male"/>';

**echo** ' Female<input type = "radio" name = "gender" value = "female"/><br/>';

**echo** ' <label for = "date">Date</label><br/>';

**echo** ' <input type = "date" name = "date"/><br/>';

**echo** ' <label for = "city">City</label><br/>';

**echo** ' <input type = "text" name = "city"/><br/>';

**echo** ' <label for = "state">State</label><br/>';

**echo** ' <input type = "text" name = "state"/><br/>';

**echo** ' <label for = "picture">Picture</label><br/>';

**echo** ' <input type = "file" name = "picture"/><br/>';

**echo** ' <label for = "captcha">Captcha</label><br/>';

**echo** " <img src = 'captcha.php' alt = 'Captcha Image'/>";

**echo** ' <input type = "text" name = "captcha"/><br/>';

**echo** ' <input type = "submit" name = "submit" value = "Submit"/><br/>';

**echo** ' </form>';

**echo** ' </body>';

**echo** ' </html>';

**require\_once**("templates/pagefooter.php");

**endif**;

?>

Code 57: signup.php (Updated Data Control Sign Up Form)

#### Graph Data

Best way to extrapolate a graph is make a two-dimensional array.

<?php

$persons\_info = [

"Samith" =>

[

"Physics" => 70,

"Chemistry" => 30,

"Math" => 50,

"Biology" => 90,

"English" => 54,

],

"Photon" =>

[

"Physics" => 10,

"Chemistry" => 40,

"Math" => 20,

"Biology" => 80,

"English" => 70,

],

"Erfan" =>

[

"Physics" => 100,

"Chemistry" => 49,

"Math" => 83,

"Biology" => 76,

"English" => 23,

],

];

**foreach** ($persons\_info **as** $person\_info => $subjects){

**echo** "<strong>".$person\_info."</strong><br/>";

**foreach**($subjects **as** $subject => $score){

**echo** $subject.": ".$score."<br/>";

}

}

?>

Code 58: multidimensionalarray.php

#### Graphing Concept

Assuming, there are **four data**

Height

1 2 3 4 5 6 7 8 9

Width

Each **star** represents the **bar width of the data** and **other spaces** are just empty spaces in between two data (bar width/stars). Therefore, each **bar width** should be

Interpreting the equation, thought process

1. We divide the width by the total number of data,
2. We need space between each data which should be the same width as the bar width
3. Also, we need an extra space at the end

Looping through the data to display the data

|  |  |  |  |
| --- | --- | --- | --- |
| **Loop** | **Bar Position** | **Relative Position** | **Breakdown with Loop** |
| *0* | 2 | x | x + (2 \* *0* \* x) |
| *1* | 4 | x + 3x | x + (2 \* *1* \* x) |
| *2* | 6 | x + 5x | x + (2 \* *2* \* x) |
| *3* | 8 | x + 6x | x + (2 \* *3* \* x) |

Therefore, the starting positions of the bars would be

|  |
| --- |
| x + (2 \* *0* \* x) |
| x + (2 \* *1* \* x) |
| x + (2 \* *2* \* x) |
| x + (2 \* *3* \* x) |

The ending positiong of the bars would be

|  |
| --- |
| 2x + (2 \* *0* \* x) |
| 2x + (2 \* *1* \* x) |
| 2x + (2 \* *2* \* x) |
| 2x + (2 \* *3* \* x) |

(0, 0)

Height

1 2 3 4 5 6 7 8 9

**(Max Height)**

The division of each height is

For a given data the height of the bar is,

To give border, we use width -1 and height -1

For imagestring, we use the same concept as the bar width, make sure that the string is above the height, therefore it will be height – 2.

<?php

**require\_once**("bargraph.php");

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "classroom";

$showform = **TRUE**;

$students = **array**();

$subjects = **array**();

**if**($showform):

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head> ";

**echo** " <title>Classroom</title>";

**echo** " <meta charset = 'UTF-8'/>";

**echo** " </head>";

**echo** " <body>";

**echo** " <form method = 'post' action = '".$\_SERVER['PHP\_SELF']."'>";

**echo** " <label for = 'search'> Search </label><br/>";

$value = (!**empty**($\_POST['search']))? $\_POST['search'] : "";

**echo** " <input type = 'text' name = 'search'"

." value = '**$value**'/><br/>";

**echo** " <input type = 'submit' name = 'submit' value = 'Search'/>";

**echo** " </form>";

**echo** " </body>";

**echo** " </html>";

**endif**;

**if**(isset($\_POST['submit'])):

*// $showform = FALSE;*

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die** ("Server Connection Denied");

$search = $\_POST['search'];

$query = "SELECT per.person\_name, sub.subject\_name, sc.score FROM score AS sc

INNER JOIN person AS per

ON sc.person\_id = per.person\_id

INNER JOIN subject as sub

ON sc.subject\_id = sub.subject\_id

WHERE per.person\_name LIKE '%**$search**%' LIMIT 5";

$result = mysqli\_query($connection, $query) **or** **die**("Query Denied");

**echo** "<table>";

**while**($row = mysqli\_fetch\_array($result)){

**if**(!in\_array($row['person\_name'], $students)):

**echo** "<tr>";

$students[] = $row['person\_name'];

**echo** "<th align = 'left'> Student: ".$row['person\_name']."</th>";

**echo** "</tr>";

**endif**;

**echo** "<tr>";

**echo** "<td> Subject: ".$row['subject\_name']."</td>";

**echo** "<td> Score: ".$row['score']."</td>";

**echo** "</tr>";

$subjects[] = [$row['subject\_name'],$row['score']];

}

**echo** "</table>";

**echo** "<pre/>";

**endif**;

draw\_bar\_graph(480, 240, $subjects, 100, "image.png");

**echo** "<img src = 'image.png' alt = 'bar graph'/>";

?>

Code 59: charting.php

<?php

**function** draw\_bar\_graph($width, $height, $data, $max\_value, $filename){

$image = imagecreatetruecolor($width, $height);

$bg\_color = imagecolorallocate($image, 255, 255, 255);

$text\_color = imagecolorallocate($image, 0, 255, 0);

$bar\_color = imagecolorallocate($image, 255, 0, 0);

$border\_color = imagecolorallocate($image, 0, 0, 255);

imagefilledrectangle($image, 0, 0, $width, $height, $bg\_color);

imagerectangle($image, 0, 0, $width - 1, $height - 1, $border\_color);

$bar\_width = $width/((count($data) \* 2) + 1);

**for**($i = 0; $i < count($data); $i++){

imagefilledrectangle($image, ($i \* $bar\_width \* 2) + $bar\_width,

$height, ($i \* $bar\_width \* 2) + ($bar\_width \*2),

$height - (($height/$max\_value) \* $data[$i][1]), $bar\_color);

imagestringup($image, 5, ($i \* $bar\_width \* 2) + $bar\_width, $height - 5, $data[$i][0], $text\_color);

}

imagepng($image, $filename, 5);

imagedestroy($image);

}

?>

Code 60: bargraph.php

### Web Services

Pushing web content to users is a great way to help gain more exposure for a website. HMTL is for viewing. **RSS** is for **syndicating**. An RSS view on a particular set of data is called an **RSS Feed.** To view an RSS Feed, a person needs RSS **newsreader.** RSS is a markup language used to describe web content for syndication. RSS is based on general markup language XML language. **RSS** stands for Really Simple Syndication.

**XML** is a markup language used to describe any kind of data.

**Ground rules for XML language**

1. Tags that contain content must appear as matching pairs. <p> Phone Home! </p>
2. Empty tags that have no content must be coded with a space and a forward slash at the end before before the closing brace. <br />
3. All attributes values must be enclosed in double quotes.

An RSS newsreader is designed to consume the data made available by an RSS newsfeed. XML code consists of tags what are also sometimes referred to as elements that form parent-child relationships.

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

**<rss** version = "2.0"**>**

**<channel>**

**<title></title>**

**<link></link>**

**<description></description>**

**<language></language>**

**<item>**

**<title></title>**

**<link></link>**

**<pubDate></pubDate>**

**<description></description>**

**</item>**

**</channel>**

**</rss>**

Code 61: rss.xml

#### Dynamically Generate XML

To make the RSS feed run in the Google Chrome Browser, we need add an extension



Fig 27: RSS Subscription Extension by Google

A standard icon is available to make it clear to users that offer an RSS Newsfeed

<?php header("Content-Type: text/xml"); ?>

<?php **echo** '<?xml version = "1.0" encoding = "UTF-8"?>'; ?>

<?php **echo** '<rss version = "2.0">'; ?>

<?php **echo** ' <channel>'; ?>

<?php **echo** ' <title>Job Search</title>'; ?>

<?php **echo** ' <link>http://sphotonkhan.com</link>'; ?>

<?php **echo** ' <description>Get to know about the hot jobs</description>'; ?>

<?php **echo** ' <language>en-us</language>'; ?>

<?php

$server\_name = "localhost";

$user\_name = "root";

$password = "";

$db\_name = "riskyjobs";

$connection = mysqli\_connect($server\_name, $user\_name, $password, $db\_name)

**or** **die** ("Server Connection Denied");

$query = "SELECT \* FROM riskyjobs ORDER BY date\_posted DESC";

$result = mysqli\_query($connection, $query)

**or** **die**("Query Denied");

**while**($row = mysqli\_fetch\_array($result)){

**echo** ' <item>';

**echo** ' <title>'.$row['title'].'</title>';

**echo** ' <link>'.$row['company'].'</link>';

**echo** ' <pubDate>'.$row['date\_posted'].'</pubDate>';

**echo** ' <description>'.substr($row['description'], 0, 50).'</description>';

**echo** ' </item>';

}

**echo** ' </channel>';

**echo** ' </rss>';

?>

Code 62: newsfeed.php

#### REST and XML

Pulling content from another site to place it on our site. Syndicating videos from YouTube involves issuing requests and handling responses. **YouTube** videos expect videos to be queried through the use of a **REST request**. Rest stands from **Representational State Transfer**. The idea is web resources should be accessible through unique links, which means we should be able to access “RESTful” data simply by constructing a URL for it. GET is the REST ‘action’ used to access the resource. Youtube reponds to video requests XML data that describes the video.

The **simplexml\_load\_file($file)** function converts the specified XML file into a SimpleXMLElement object.

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

**<note>**

**<to>**Elizabeth**</to>**

**<from>**Monkey King**</from>**

**<heading>**Party**</heading>**

**<body>**Don't forget to bring the chocolates!**</body>**

**</note>**

Code 62: note.xml

<?php

$note = simplexml\_load\_file("note.xml");

*//Accessing the value to w.r.t tags*

**echo** $note->to."<br/>";

**echo** $note->heading."<br/>";

**echo** $note->body."<br/>";

**echo** $note->from."<br/><br/>";

*//Accessing tags*

**echo** $note->getName()."<br/>";

**foreach**($note->children() **as** $child){

**echo** $child->getName().": ".$child."<br/>";

}

?>

Code 63: xmlload.php

When we an XML tag that has two names separated by a colon, we are looking at **namespace.** Namespaces are named groups of XML tags, while entities are used to encode special characters within XML documents.

XML entity, symbolic way of referencing a special character, such as &, < or >

&amp; = &

&lt; = <

&gt; = >

&quot; = “

&apos; = ‘

An element is just an abstract way of thinking of an XML tag and the data it contains. The XML data is organized into hierarchy of elements (tags)

A **PHP object** is a special data type that allows data to be packaged together with functions in a single construct. Objects also have **methods,** which are functions that are tied to an object. To access the data we use object **properties**, which are individual pieces of data stored within an object. XML document is a **collection of objects**. Namespaces make it a bit trickier to access elements within XML data. Use the children() method to isolate all elements associated with a namespace.

### Database Privileges

You can set very specific user privileges, even control what our user can do to a specific column

CREATE USER photon IDENTIFIED BY ‘khan\_photon’

GRANT SELECT, INSERT ON table\_name TO photon

### SQL Error Check

mysqli\_error($connection) gives us a clue as to exactly what went wrong.

### Sanitize External Data

strip\_tags() removes any html tags from a string.

### Precedence



### Handling Errors

**readfile("web.txt")**

The readfile() function is useful if all we want to do is open up a file and read its contents.

**fopen(file\_path, “r”)**

It contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

**fread($myfile,filesize("web.txt"))**

First parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

**fclose($file)**

This function is used to close an open file.

**fgets($file)**

This function is used to read a single line from a file.

**feof($file)**

This function checks if the "end-of-file" (EOF) has been reached.

It is useful for looping through data of unknown length.

**fwrite($file, $string)**

It contains the name of the file to write to and the second parameter is the string to be written.

AJAX = Asynchronous JavaScript and XML

CSS = Cascading Style Sheets

HTML = Hyper Text Markup Language

PHP = PHP Hypertext Preprocessor

SQL = Structured Query Language

SVG = Scalable Vector Graphics

XML = EXtensible Markup Language

Code 64: web.txt

<?php

*//echo readfile("web.txt");*

**try**{

$file = fopen("web.txt", "r");

*/\*\**

*\* This reads out all text in one line*

*\*/*

*// echo fread($file, filesize("web.txt"));*

*/\*\**

*\* Searchs for the end of line and follows the format*

*\*/*

**while**(!feof($file)){

**echo** fgets($file)."<br/>";

}

}

**catch**(Except $e){

**echo** "Message: ".$e->getMessage();

}

**finally**{

fclose($file);

}

?>

Code 65: read.php

<?php

**try**{

$file = fopen("new.txt", "w");

$txt = "Damn sone **\n**";

fwrite($file, $txt);

$txt = "You crazy, Bro!**\n**";

fwrite($file, $txt);

}

**catch**(exception $e){

**echo** "Error: ".$e->getMessage();

}

**finally**{

fclose($file);

}

?>

Code 66: write.php

### PHP Terms

**mail($to, $subject, $message, 'From:'.$email)**

Send email

**$conn = mysqli\_connect($server\_name, $username, $password, $database\_name)**

Connect PHP script with the server

**mysqli\_close($conn)**

Closes the connection with the server

**mysqli\_query($conn, $query)**

Sends MySQL Query to the server

**mysqli\_select\_db($connection, $databasename)**

It is used to change the default database for the connection

$**result = mysqli\_query($conn, $query)**

**mysqli\_fetch\_array($result)**

It retrieves a single row of data from the results of a database query

**isset()**

Variable exists or not, any value is assigned to it or not

**empty()**

Checks whether the variable contains 0, NULL, FALSE or an empty string

**include(“file\_path”), include\_once(“file\_path”)**

**require(“file\_path”), require\_once(“file\_path”)**

Share script code across multiple files.

**$\_FILES**

It is a built-in super global variable provides access to information about uploaded files.

**$\_FILES[‘image’][‘name’]**

The name of the uploaded file

**$\_FILES[‘image’][‘type’]**

MIME type of the uploaded file

**$\_FILES[‘image’][‘tmp\_name’]**

Temporary storage location of the file on the server

**$\_FILES[‘error’][‘error’]**

The error code for the file upload, 0 indicates a success, other values indicate failure

**move\_uploaded\_file($\_FILES[‘image’][‘tmp\_name’], $target)**

Move file from temporary location to the targeted location

**trim($POST[‘name’])**

Eliminates Leading or trailing spaces

**mysqli\_real\_escape\_string($connection, trim($\_POST[‘name]));**

It converts dangerous characters into an escaped format that won’t adversely affect SQL queries.

**header(’Location:http://www.addinventory.php’);**

The header is called a location header and redirects the current page to addinventory.php

**header(’Refresh: 5; url=http://www.addinventory.php’);**

This header is called a refresh header since it refreshes a page after a period of time.

**header(’Content-Type: text-plain’);**

Context will be plain text

**header('HTTP/1.1 401 Unauthorized');**

This header lets the browser know that the user in not authorized to view the page

**header('WWW-Authenticate: Basic realm="Store"');**

Basic realm is just a phrase used to uniquely identify this. Particular authentication-it appears in the authentication window.

**setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/");**

The cookie will expire after 30 days (86400 \* 30). The "/" means that the cookie is available in entire website

**dirname($\_SERVER['PHP\_SELF'])**

/education/Php/7. Personal Web Application

**$\_SERVER['HTTP\_HOST']**

Localhost

**session\_start(), session\_destroy(), session\_name()**

Checks the session id

**array\_push($array, $items)**

It tacks a new element onto the end of an array, causing the array to grow by one

**array\_slice(array,start,length,preserve)**

Starts the array from the start the position

**explode(‘delimiter’, ‘phrase’)**

Delimiter separates the words in the phrase

**implode(‘delimiter’, ‘items’)**

Combines the string in the items array by adding delimiter after each item

**str\_replace(‘delimiter’, ‘replace\_item’, $string);**

The substring ‘delimiter’ gets replaced by ‘replace item’ in the string

**substr(string, start, length)**

The PHP function **substr()** function allows us to extract a portion of a string**.**

**ceil($total\_query/$results\_per\_page)**

This rounds up the value

**preg\_match($regex, $my\_string)**

This function takes a regex pattern and a text string, it returns true if it matches

**checkdnsrr($domain)**

This function checks whether the domain is valid or not.

**rand()**

This function returns a random integer within a certain range.

**chr()**

This build-in function converts a number to its ASCII character equivalent.

**$image = imagecreatetruecolor($width, $height)**

The function creates a blank image in memory ready to be drawn to with other GD functions. This is an **image identifier**. Then to change the background color before drawing anything we need to call

**imagefilledrectangle($image, x1, y1, x2, y2, $color)**

It draws a rectangle whose interior is filled with the specified color.

**imagecolorallocate($image, red, green, blue)**

It is **color identifier**

**imagesetpixel($image, rand(0, $width), rand(0, $height), $graphic\_color)**

The function draws a single pixel at a specified coordinate within the image. Coordinates start at 0,0 in the upper left corner of the image and increase to the right and down.

**imageline($image, 0, rand(0, $height), rand(0, $width), rand(0, $height), $graphic\_color)**

This function draw a line between between two co-ordinates(x1, y1 and x2, y2)

**imagefilledellipse(image, cx, cy, width, height, color)**

For drawing circles and ellipses, this function accepts a center point and a width and height. A perfect circle is just an ellipse with an equal height and width

**imagefilledarc(image, cx, cy, width, height, start, end, color, style)**

**imagefilledpolygon(image, points, num\_points, color)**

These function works similar like the ellipses one

**imagepng(image, filename, 5)**

This function returns true or false depending on whether the image was successfully created.

Writes the image to a PNG file with the specified filename and a compression level of 5

**imagedestroy($image)**

Always free up images in memory with imagedestroy() one we have output them

**imagettftext(image, size, angle, x, y, color, fontfile, text)**

Use the imagettftext() function to draw highly customized text with our own TrueType font.

**imagegif(image)**

It is used to create gif image

**imagejpeg(image)**

It is used to create jpeg image

**imagecolortranparent($image)**

This sets color as a transparent color with an image. This must be a color that we have created using imagecolorallocate(). To generate the image, just either imagegif() or imagepng(). image(jpeg) doesn’t support transparency

**sha1(str) or md5(str)**

PHP offers encryption

**unlink($file) + @ = @unlink(file)**

It deletes a file from the web server. We can suppress error reporting with @ in case the file upload didn’t actually happen.

The **simplexml\_load\_file($file)** function converts the specified XML file into a SimpleXMLElement object.

**mysqli\_error($connection)**

gives us a clue as to exactly what went wrong.

**strip\_tags()**

removes any html tags from a string.

**readfile("web.txt")**

The readfile() function is useful if all we want to do is open up a file and read its contents.

**fopen($file\_path, “r”)**

It contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

**fread($myfile,filesize("web.txt"))**

First parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

**fclose($file)**

This function is used to close an open file.

**fclose($file)**

This function is used to close an open file.

**fgets($file)**

This function is used to read a single line from a file.

**feof($file)**

This function checks if the "end-of-file" (EOF) has been reached.

It is useful for looping through data of unknown length.

**fwrite($file, $string)**

It contains the name of the file to write to and the second parameter is the string to be written.

### SQL Terms

DROP TABLE table\_name

DESCRIBE table\_name

SELECT \* FROM table\_name

DELETE FROM table\_name

ALTER TABLE table\_name ADD COLUMN column\_name column\_type

ALTER TABLE table\_name CHANGE COLUMN old\_column new\_column new\_column\_type

ALTER TABLE table\_name DROP COLUMN column\_name

ALTER TABLE table\_name MODIFY COLUMN column\_name column\_type AFTER column\_nam

SELECT \* FROM table\_name ORDER BY column ASC/DESC

DELETE FROM table\_name WHERE COLUMN = match LIMIT num

INSERT INTO table\_name(column\_name…) VALUES(column\_values…)

UPDATE table\_name SET column\_name = some\_value WHERE column\_name = some\_value

SELECT mismatch\_topic.topic\_id, mismatch\_category.name

FROM mismatch\_topic

INNER JOIN mismatch\_category

USING (category\_id)

WHERE mismatch\_topic.name = “Easy Listening Music”

CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);

ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

ALTER TABLE Orders  
DROP FOREIGN KEY FK\_PersonOrder;

SELECT job\_id title, description FROM jobs WHERE title LIKE ‘%fighter%’

SELECT FROM table\_name WHERE COLUMN = match LIMIT num(skip) num(show)

Encrypt data using MD5() in my MySQL

SHA() is considered a little secure more than MD5()

CREATE USER photon IDENTIFIED BY ‘khan\_photon’

GRANT SELECT, INSERT ON table\_name TO photon

### Homework

1. Combine HTML(form.html) and PHP file(report.php) into one PHP file (form.php)
2. Edit (addemail.php, addemail.html) into one PHP file(addemailupdated.php) and this time only unique email addresses would be added to the database.
3. Edit (filevalidation.php) to make it more robust.
4. Add approve and unapproved to the store files.
5. Add Validation to the Personal Web Application files.
6. Change the authentication from cookies to session for personal web application.
7. Revamp the templates for the personal web application
8. Create a REGEX for email and NID number
9. Create a Form with validation and captcha

## PHP Certification

### Basics

PHP’s syntax is derived from many languages – predominantly the C language then Perl, Java. PHP is primarily designed as a text processor. PHP can be inserted directly into a text file using a special set of tags; the interpreter will then output any text outside the tags as-is and execute the code that is between

the tags

#### Types of Tags

|  |  |
| --- | --- |
| Standard Tags | **<?php … code … ?>** |
| Short Tags | **<? … code … ?>** || **<?= $variable ?>** |
| Script Tags | **<script language = “php”> …code … </script>** |
| ASP Tags | **<% … code … %>** |

**Standard tags** are the are the best solution for portability and backwards compability

**Short tags** have major drawback of conflicting with XML headers

**Script tags** can also be used but however it ignores the code in standard tags

**Short tags, Script tags and ASP tags are all considered deprecated**

**echo** is not a function and it does not have a return value.

We can out data through a function, which is **print(),** it has a **return value of 1**.

**die(),** is an alias of exit() It allows us to terminate the scripts output.

#### PHP Data Types

PHP support smany different data types, they are divided into two categories: ***scalar and composite***.

A scalar value contains only one value at a time. PHP support four scalar types

**boolean, int, float, string**

There are two types of numbers, **integers and floating-point** values. Numbers can be declared using several different **notations**.

**Subsection of INT numbers (Notations: Decimal, Octal, Hexadecimal)**

Calculating *Octal to Decimal*

Calculating *Hexadecimal to Decimal*

Ox: It means the following number is hexadecimal,

**Subsection of Float-point values (Notations: Decimal, Exponential)**

They are also called floats and sometimes doubles.

Converting *Exponential to decimal*

**Strings** are ordered collection of binary data (text, music, spreadsheet, image file)

**Booleans** can only be written in two values: TRUE or FALSE;

**Compound Data Types**

They are usually the container of data types

**Arrays** are containers of ordered data elements

**Objects** are containers of both data and code.

**Other Data Types**

**NULL** indicates the variable has no value.

**Resource** indicates external resouces that are not used natively by PHP

$fp = fopen("index.php",'r');

$conn = mysqli\_connect(localhost,"root","admin","animals");

**Converting Data types**

Enclosed in brackets and placed before an expression

<?php

*//Octal Number*

$a = 0123;

**echo** $a."<br/>";

*//Hexadecimal Number*

$a = 0x1A;

**echo** $a."<br/>";

*//Exponential*

$a = 2E7;

**echo** $a."<br/>";

*//Casting*

$x = 10.88;

**echo** (int) $x;

?>

**Code 1: numbernotation.php**

**Variables**

PHP is loosely typed, meaning that it will implicity change the type of a variable as needed

**$name,** ‘valid’;

**$\_name,** ‘valid’;

**$1name,** ‘invalid’;

**Variable Variables**

<?php

$name = "full\_name";

$$name = "Photon Khan";

**echo** $full\_name."<br/>";

$number = '123';

$$number = '456';

**echo** ${'123'}."<br/>";

**function** display($text){

**echo** $text."<br/>";

}

$function\_name = 'display';

$function\_name("I am a disco dancer");

?>

**Code 2: variablevariable.php**

#### Bitwise Operator

This operator allows us to manipulate bits of data. They are designed to work only on integers.

|  |  |
| --- | --- |
| $a && $b | **Logical AND** between a and b (BOOLEAN) |
| $a || $b | **Logical OR** between a and b (BOOLEAN) |
| $a & $b | **Bitwise AND** between a and b |
| $a | $b | **Bitwise OR** between a and b |
| $a ^ $b | **Bitwise XOR** between a and b |
| ~ $a | **Bitwise NOT** of a |
| $a << $b | **Shift left** -- Shift the bits of $a $b steps to the left (each step means "multiply by two") |
| $a >> $b | **Shift right** -- Shift the bits of $a $b steps to the right (each step means "divide by two") |

**Signed variables**, such as signed integers will allow you to represent numbers both in the positive and negative ranges.

**Unsigned variables**, such as unsigned integers, will only allow you to represent numbers in the positive.

Binary Representation

|  |  |  |
| --- | --- | --- |
| **Decimal** | **Binary** | **Hexadecimal** |
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0010 |
| 3 | 0011 | 0011 |
| 4 | 0100 | 0100 |
| 5 | 0101 | 0101 |
| 6 | 0110 | 0110 |
| 7 | 0111 | 0111 |
| 8 | 1000 | 1000 |
| 9 | 1001 | 1001 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

AND

|  |  |  |  |
| --- | --- | --- | --- |
| **Input(a)** | **Input(b)** | **Output(y)** | **Boolean** |
| 0 | 0 | 0 | FALSE |
| 0 | 1 | 0 | FALSE |
| 1 | 0 | 0 | FALSE |
| 1 | 1 | 1 | TRUE |

OR

|  |  |  |  |
| --- | --- | --- | --- |
| **Input(a)** | **Input(b)** | **Output(y)** | **Boolean** |
| 0 | 0 | 0 | FALSE |
| 0 | 1 | 1 | TRUE |
| 1 | 0 | 1 | TRUE |
| 1 | 1 | 1 | TRUE |

NOT

|  |  |  |
| --- | --- | --- |
| **Input(a)** | **Output(y)** | **Boolean** |
| 0 | 1 | TRUE |
| 1 | 0 | FALSE |

XOR

|  |  |  |  |
| --- | --- | --- | --- |
| **Input(a)** | **Input(b)** | **Output(y)** | **Boolean** |
| 0 | 0 | 0 | FALSE |
| 0 | 1 | 1 | TRUE |
| 1 | 0 | 1 | TRUE |
| 1 | 1 | 0 | FALSE |

Most computers use two's-complement arithmetic**. Negative numbers** are created by taking the one's-complement (flip all the bits) and adding one:

Converting -5 to binary

|  |  |
| --- | --- |
| 5 (Decimal) | 0101 (binary) |
| 1’s complement | 1010 (Flip) |
| Add 1 | 1011 (Adding) {According to Computer) |

**Signed byte** holds values from **-128 to +127**, the first bit indicates whether the value is negative or positive.

~7(decimal) 🡪 ~0111(binary) 🡪 1000 (first bit 1 means negative) 🡪 -8

**Unsigned byte** holds values from **0 to 255**

*PHP do not support the unsigned integers*

Bitwise Math

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Bitwise AND** | | | | **Bitwise OR** | | |
| 3 | & | 7 | | 3 | | | 7 |
| 0011 | & | 0111 | | 0011 | | | 0111 |
| 0011  0111  0011 🡪 3 | | | | 0011  0111  0111 🡪 7 | | |
| **Bitwise XOR** | | | |
| 3 | ^ | 7 |
| 0011 | ^ | 0111 |
| 0011  0111  0100 🡪 4 | | | |

|  |
| --- |
| **Bitwise NOT** |
| ~7  ~0111  1000 (Flip)  -8 (signed)  8 (unsigned) |

Bitwise NOT -5(~-5) 🡪 1011(Flip) 🡪 0100 🡪 4 (in decimal)

Shifting Operators

(Left Shift) (Multiply by the power of two)

3 ≪ 1

3 in binary 🡪 0011

3 shifts by 1 to the left 🡪 0110 🡪 6 (in decimal)

(Right Shift) (Multiple by the power of two)

3 ≫ 1

3 in binary 🡪 0011

3 shifts by 1 to the right 🡪 0001 🡪 1 (in decimal)

**Use** **pow(number, power)**

2^3 = 8

<?php

$a = 3;

$b = 7;

$c = -5;

$shift\_by\_one = 1;

**echo** "<!doctype html>";

**echo** " <html>";

**echo** " <head>";

**echo** " <title>Bitwise Operations</title>";

**echo** " <meta charset = 'UTF-8'>";

**echo** " </head>";

**echo** " <body>";

**echo** "<table>";

**echo** " <tr>";

**echo** " <th align='left'>Operations</th>";

**echo** " <th align='left'>Solution</th>";

**echo** " </tr>";

**echo** " <tr>";

**echo** "<td>**$a** & **$b** (Bitwise AND)</td>";

**echo** "<td>".($a & $b)."</td>";

**echo** " </tr>";

**echo** " <tr>";

**echo** "<td>**$a** | **$b** (Bitwise OR)</td>";

**echo** "<td>".($a | $b)."</td>";

**echo** " </tr>";

**echo** " <tr>";

**echo** "<td> **$a** ^ **$b** (Bitwise XOR)</td>";

**echo** "<td>".($a ^ $b)."</td>";

**echo** " </tr>";

**echo** " <tr>";

**echo** "<td> ~**$b** (Bitwise NOT)</td>";

**echo** "<td>".(~$b)."</td>";

**echo** " </tr>";

**echo** " <tr>";

**echo** "<td> **$c** (Bitwise NOT)</td>";

**echo** "<td>".(~$c)."</td>";

**echo** " </tr>";

**echo** " <tr>";

**echo** "<td> **$a** << **$shift\_by\_one** (Bitwise Left Shift)</td>";

**echo** "<td>".($a << $shift\_by\_one)."</td>";

**echo** " <tr>";

**echo** "<td> **$a** >> **$shift\_by\_one** (Bitwise Right Shift)</td>";

**echo** "<td>".($a >> $shift\_by\_one)."</td>";

**echo** " </tr>";

echo " <tr>";

echo "<td> 2 to the power of 3 </td>";

echo "<td> ".pow(2,3)." </td>";

echo " </tr>";

**echo** "</table>";

**echo** " </body>";

**echo** " </html>";

?>

**Code 3: bitwise.php**

1 ≪ 32 in 32-bit computer gives a value of 0

#### Referencing Variables

<?php

*//Referencing variables*

$a = 10;

**echo** nl2br("Before Changing: **$a\n**");

$b = &$a;

$b = 20;

**echo** nl2br("After Changing: **$a\n**");

*//Referencing variables through function*

$ready\_to\_change = 4;

**echo** nl2br("Before Changing: **$ready\_to\_change\n**");

**function** change\_me(&$let\_me\_change){

$let\_me\_change = 6;

}

change\_me($ready\_to\_change);

**echo** "After Changing: **$ready\_to\_change**";

?>

**Code 4: referencing.php**

#### Comparison Operators

|  |  |
| --- | --- |
| $a == $b | Check if $a is equal to $b |
| $a === $b | Check if $a is equal to $b and their data type should be same |
| $a != $b | Chech if $a is not equal to $b |
| $a !== $b | Check if $a is not equal to $b and its data type |
| $a < $b | Check if $a is less than $b |
| $a > $b | Check if $a is greater than $b |
| $a ≤ $b | Check if $a is less or equal to $b |
| $a ≥ $b | Check if $a is greater or equal to $b |

#### ASCII Values



<?php

*//String to String Comparison*

$left = "ABC";

$right = "ABD";

*// A = 065, B = 066, C = 067, D = 068*

**if**($left > $right):

*// 065 066 067 > 065 066 068*

**echo** "TRUE"."<br/>";

**else**:

**echo** "FALSE"."<br/>";

**endif**;

*//Lower case and Upper case comparison*

**if**("apple" > "Apple"):

*// 097 065*

**echo** "TRUE"."<br/>";

**else**:

**echo** "FALSE"."<br/>";

**endif**;

?>

**Code 5: comparison.php**

#### Binary Operators

|  |  |  |
| --- | --- | --- |
| **Binary Operator** | **Alternative** | **Description** |
| $a && $b | $a and $b | Gives true when both the variables are true |
| $a || $b | $a or $b | Gives true when **0 1**, **1 0**, **1 1** |
| $a XOR $b | -------- | Gives truen when **0 1, 1 0** |

#### Other Operators

$x = @mysqli\_connect();

@ error suppression operator

$a = `ls -l`;

This allows to run the shell command

<?php

*//Works in Linux Only*

$a = `ls -l`;

**echo** $a;

*//Works in windows (To show what the command window will say)*

system("python 2>&1",$output) ;

?>

**Code 6: backtick.php**

#### Alternative Syntaxes

<?php

$bunch\_of\_numbers = [23, 42, 12, 100, 14, 29, 10];

$max\_number = 0;

*//Finding out the maximum value;*

**foreach**($bunch\_of\_numbers **as** $number):

**if**($max\_number < $number):

$max\_number = $number;

**endif**;

**endforeach**;

**echo** nl2br("Highest Number is: **$max\_number\n**");

**echo** nl2br("**\n**");

*//While Loop*

**echo** nl2br("While Loop**\n**");

$i = 0;

**while**($i < 4):

**echo** nl2br("**$i\n**");

$i++;

**endwhile**;

**echo** nl2br("**\n**");

*//For Loop*

$i = 0;

**echo** nl2br("For Loop**\n**");

**for**($i = 0; $i < 3; $i++):

**echo** nl2br("**$i\n**");

**endfor**;

**echo** nl2br("**\n**");

*//Do While (No Alternative Syntax)*

**echo** nl2br("Do While**\n**");

$i = 0;

**do**{

**echo** nl2br("**$i\n**");

}**while**($i > 0);

**echo** nl2br("**\n**");

*//Switch Case*

**echo** nl2br("Switch Case**\n**");

$items = ["Apple", "Bat", "Cat", "Date", "Fall"];

**foreach**($items **as** $item):

**switch**($item):

**case** "Cat":

**echo** nl2br("Moderate**\n**");

**break**;

**case** "Bat":

**echo** nl2br("Good Work**\n**");

**break**;

**case** "Apple":

**echo** nl2br("Excellent**\n**");

**break**;

**case** "Fall":

**echo** nl2br("Dang it!**\n**");

**break**;

**case** "Date":

**echo** nl2br("Phew!**\n**");

**break**;

**endswitch**;

**endforeach**;

?>

**Code 7: alternativesyntax.php**

<?php

*//Continue*

**echo** nl2br("Continue**\n**");

$total = 4;

**for**($i = 1; $i <= $total; $i++):

**if**($i == 2):

**continue**;

**endif**;

**echo** nl2br("**$i\n**");

**endfor**;

*//Break*

**echo** nl2br("**\n**");

**echo** nl2br("Break**\n**");

$total = 4;

**for**($i = 1; $i <= $total; $i++):

**if**($i == 2):

**break**;

**endif**;

**echo** nl2br("**$i\n**");

**endfor**;

?>

**Code 8: continue.php**

**nl2br(“\n”)**

New line syntax

**PHP\_EOL**

Create space between two numbers. EOL is End of Line

#### Types of Errors

|  |  |
| --- | --- |
| Compile-time Errors | Error while compiling (by **parser, compiler, interpreter**) |
| Fatal Errors | Halt the execution of the script |
| Recoverables errors | Failures that can be handled in a safe way |
| Warnings | Recoverable errors at run-time |
| Notices | Error condition but not significant enough to halt the script |

**Custom Error Handler**

It is useful to automatically notify the coder where the problem lies.

**set\_error\_handler("customError")**

This script can declare a catch-all function that is called by PHP when an error condition occurs

**customError(error\_level,error\_message, error\_file, error\_line, error\_ context)**

Error Message (required), it is the user defined error

Error Levels (required)

|  |  |
| --- | --- |
| 2 | Warnings |
| 8 | Notice |
| 256 | User Error |
| 512 | User Warning |
| 1024 | User Notice |
| 4096 | Recoverable Error |
| 8191 | All |

**ob\_start()**

Turn on output buffering

**ob\_get\_contents()**

Return the contents of the output buffer

**ob\_end\_clean()**

Clean (erase) the output buffer and turn off output buffering

*/\*\**

*\* [customError]*

*\* @param int $errno error number*

*\* @param string $errstr error string*

*\* @return NULL*

*\*/*

*//Show Error (Notice: 8)*

**function** customError($errno, $errstr, $errfile, $errline){

*// ob\_start();*

**echo** "Error: [**$errno**] **$errstr**, on line **$errline**";

*// $str = ob\_get\_contents();*

*// error\_log($str);*

*// ob\_end\_clean();*

*// var\_dump($str);*

error\_log("Error: [**$errno**] **$errstr**, on line **$errline**",1,

"khan.photon@gmail.com","From: khan.photon@gmail.com");

}

set\_error\_handler("customError");

**echo**($test);

**echo** "<br/>";

*//Trigger Error (User Notice: 1024)*

$i = 2;

**if**($i > 1){

trigger\_error("Value should be 1");

}

?>

**Code 9: handlingerrors.php**

**error\_log("Message",1,"$email","From: $email");**

Send Email About the Error

### Functions

#### Returning Reference

Functions can also be declared so that they return by reference; this allows us to return a variable as the result of the function. We must return a variable – we cannot return an expression by reference, or use an empty return statement to force a NULL return value

#### Variable Scope

There are three variable scope: the global scope, function scope, and class scope. There are two ways to access variables in the global scope from inside a function. global $var and $GLOBALS[‘var’]

<?php

$number = 10;

**function** &getValue(){

*// global $number;*

*// We cannot return return global $number;*

*// return $number;*

**return** $GLOBALS['number'];

}

$collect =& getValue();

--$collect;

**echo** nl2br("Returning reference: **$number**");

?>

**Code 9: returningreference.php**

PHP provides three built-in-function to handle variable-length argument lists:

**func\_num\_args()**

Get the number of arguments passed in the function

**func\_get\_arg()**

Get the one argument passed int the function

**func\_get\_args()**

Get the arguments value passed in the function

**array\_shift($array)**

Remove the first element (red) from an array, and return the value of the removed element

$a=array("a"=>"red","b"=>"green","c"=>"blue");  
**echo array\_shift($a);** Output: red  
**print\_r ($a);** Output**: (**"b"=>"green","c"=>"blue");

<?php

**function** getdata(){

**if**(func\_num\_args() == 0):

**echo** "You need to specify at least one argument";

**else**:

$args = func\_get\_args();

**switch**(func\_num\_args()):

**case** 1:

**return** "One Item: ".$args[0];

**break**;

**case** 2:

**return** "Two items: ".array\_shift($args).", ".array\_shift($args);

**break**;

**default**:

**endswitch**;

**endif**;

}

$one\_item = getData("potato");

$two\_items = getData("potato", "tomato");

**echo** nl2br("**$one\_item\n**");

**echo** nl2br("**$two\_items\n**");

?>

**Code 9: variablelengtharguments.php**

**strlen($string)**

Measure the length of the string

**<?php echo strlen("Potato");?>**

### Arrays

All arrays are ordered collections of items, ***called element.***

#### Printing Array

With arrays, echo function is not used due to its inability to deal with composite data types like arrays and objects. PHP provides two functiosn that can be used to output a variable’s value recursively: print\_r() and var\_dump().

|  |  |
| --- | --- |
| **print\_r()** | **var\_dump()** |
| Prints out the data from the array | Prints out data with its data type for each value |
| Prints only for one variable | Capable of outputting the value of more than one variable. |
| It can return its output as a string | It outputs scripts standard output |
| **Both of them offer a more specialized set of functionalities as an aid in debugging** | |

#### Enumerative vs Associative

Arrays can be divided into in two categories: enumerative and associative.

**Enumerative arrays** are indexed using only numerical indexes.

$collection = [eat, sleep, drink]; 0 🡪 eat, 1 🡪 sleep, 2 🡪 drink

$collection[0] 🡪 eat

$collection[1] 🡪 sleep

$collection[2] 🡪 drink

**Associative arrays** are sometimes referred to as dictionaries.

$collection = [‘name’ => ‘Sam’, ‘phone’ => ‘123’]

$collection[‘name’] 🡪 ‘sam’;

$collection[‘phone’] 🡪 ‘123’;

PHP automatically assigns a numberic one that is equal to the greatest numeric key already in existence in the array plus one.

<?php

$employees = [

'Photon' =>[

'id' => '433',

'occupation' => 'engineer'

],

*//Assuming we couldn't get the name*

'0' => [

'id' => '512',

'occupation' => 'Technician'

],

'Sam' => [

'id' => '102',

'occupation' => 'Human Resource'

],

];

**foreach**($employees **as** $employee => $datas):

**echo** nl2br("<strong>**$employee**</strong>**\n**");

**foreach**($datas **as** $information => $data):

**echo** nl2br("**$information**: **$data\n**");

**endforeach**;

**echo** nl2br("**\n**");

**endforeach**;

$employees[] = 'Brianna';

**echo** "<strong>".$employees[1]."</strong>";

**echo** "<pre>";

var\_dump($employees);

**echo** "</pre>";

?>

**Code 9: associative.php**

#### Array Operations

A number of operators behaves differently ently if their operarands are arrays. If two arrays have the same common elements that also share the same string keys or that have numeric keys, they would only appear once in the end.

<?php

$collection\_one = [1, 2, 3];

$collection\_two = ["a" => 1, "b" => 2, "c" => 3];

*/\*\**

*\* Therefore the addition of $collection\_one + $collection\_one:*

*\* 0 --> 1, 1 --> 2, 2 --> 3*

*\*/*

**echo** "<pre>";

var\_dump($collection\_one + $collection\_one);

**echo** "</pre>";

*/\*\**

*\* Therefore the addition of $collection\_one + $collection\_two:*

*\* 0 --> 1, 1 --> 2, 2 --> 3, a --> 1, b --> 2, c --> 3*

*\*/*

**echo** "<pre>";

var\_dump($collection\_one + $collection\_two);

**echo** "</pre>";

*/\*\**

*\* Therefore the addition of $collection\_one + $collection\_two:*

*\* 0 --> 1, 1 --> 2, 2 --> 3, a --> 1*

*\*/*

$collection\_one = [1, 2, 3];

$collection\_two = ['a' => 1, 2, 3];

**echo** "<pre>";

var\_dump($collection\_one + $collection\_two);

**echo** "</pre>";

?>

**Code 10: arrayoperations.php**

#### Comparing Arrrays

The **equivalence operator** == returns true if both arrays have the same number of elements with the same values and keys, regardless of their order.

**Identity operator** === returns true only if the array contains the same key/value pairs in the same order.

<?php

$a = [1, 2, 3];

$b = [1 => 2, 2 => 3, 0 => 1];

$c = ['a' => 1, 'b' => 2, 'c' => 3];

**echo** "<pre>";

var\_dump($a == $b);

**echo** "<pre/>";

**echo** "<pre>";

var\_dump($a === $b);

**echo** "<pre/>";

**echo** "<pre>";

var\_dump($a == $c);

**echo** "<pre/>";

**echo** "<pre>";

var\_dump($a === $c);

**echo** "<pre/>";

?>

**Code 11: comparingarrays.php**

#### Counting, Searching and Deleting Elements

**count($array)**

It counts the number of elements in an array

**is\_array($array)**

Checks whether the variable contains array or not

**isset()** has the major drawback of considering an element whose value is NULL, Therefore, it cannot but used to figure out whether an array with a given key exists or not. Instead use **array\_key\_exists()**

**unset($key)**

An element can be deleted from an array by unsetting it.

<?php

$collection = ["Sam" => "Fat", "Peter" => "thin", "Robert" => **NULL**];

**echo** nl2br(count($collection)."**\n**");

**echo** "<pre>";

var\_dump(is\_array($collection));

**echo** "</pre>";

**echo** "<pre>";

var\_dump(isset($collection['Robert']));

**echo** "</pre>";

**echo** "<pre>";

var\_dump(array\_key\_exists("Robert", $collection));

**echo** "</pre>";

unset($collection['Peter']);

**echo** "<pre>";

var\_dump($collection);

**echo** "</pre>";

?>

**Code 12: countsearchdelete.php**

#### Flipping and Reversing

**array\_flip($array)**

Inverts the value of each elements of an array with its key

**array\_reverse($array)**

Reverses the order of the elements in an array. Key association is only lost for those whose key association are numeric.

<?php

$a = ["Neville" => "Smart", "Harry" => "Intelligent", "Ron" => "Dumb"];

*// Smart --> Neville, Harry --> Intelligent, Dumb --> Ron*

*// Key becomes the value and the value becomes the key*

**echo** "<pre>";

var\_dump(array\_flip($a));

**echo** "</pre>";

$a = ["Neville" => "Smart", "10" => "Intelligent", "Ron" => "Dumb"];

*//Reversing the order*

**echo** "<pre>";

var\_dump(array\_reverse($a));

**echo** "</pre>";

?>

**Code 12: flippingreversing.php**

#### Array Pointer

Each array has a pointer that indicates the “current” element of an array in an iteration.

**reset($array)**

It reset the pointer to its intial position of an array

**prev($array)**

Move backward in an array

**next($array)**

Move forward in an array

**current($array)**

Current element in an array

**key($array)**

Current element’s key

**end($array)**

Start the iteration from the end of the array

<?php

$books = ["Da Vinci Code" => "Dan Brown", "Harry Potter" => "J.K Rowling", "John Oliver" => "Charles Dickens"];

reset($books);

**while**(key($books) != **NULL**):

**echo** "Book Name:: ".key($books)."<br/>";

**echo** "Author Name:: ".current($books)."<br/><br/>";

next($books);

**endwhile**;

**echo** nl2br("**\n**");

reset($books);

end($books);

**while**(key($books) != **NULL**):

**echo** "Book Name:: ".key($books)."<br/>";

**echo** "Author Name:: ".current($books)."<br/><br/>";

prev($books);

**endwhile**;

?>

**Code 12: arraypointers.php**

#### Referencing array in a loop

Modifying the contents of the array directly by assigning the value of each element to the iterated variable by reference rather than by value.

<?php

*//Modify the values of the all the elements*

$values = **array**("100", "200", "300");

**foreach**($values **as** $key => &$value):

$value++;

**endforeach**;

**echo** "<pre>";

var\_dump($values);

**echo** "</pre>";

*//Modify all the elements except the last one*

$values = **array**("100", "200", "300");

**foreach**($values **as** $key => &$value):

**endforeach**;

*//Loop Result 0 --> 100, 1--> 200, 2 --> 300*

**echo** "<pre>";

var\_dump($values);

**echo** "</pre>";

**foreach**($values **as** $key => $value):

**endforeach**;

*//Loop Result 2 --> 100, 2 --> 200, 2 --> 200*

**echo** "<pre>";

var\_dump($values);

**echo** "</pre>";

?>

**Code 12: arrayloopvaluereference.php**

*//Modify all the elements except the last one*

As soon as the second loop starts, $values is now assigned the value of each element. However, $v is already a referene to $values[2]; therefore, any value assigned to it will ve copied automatically into the last element of the arrays.

#### Passive Iteration

**array\_walk()** and **array\_walk\_recursive()** can be used to perform an iteration of an array in which user-defined function is called.

**array\_combine($keys, $values)**

Creates an array by using one array for keys and another for its values

**strtoupper($var)**

Changes lower case letter to uppercase

<?php

$employees = ["Peter","James"];

$description[] = ["Spiderman", "Photographer"];

$description[] = ["Guitar Player", "Metallica"];

$combine = array\_combine($employees, $description);

**function** setCase(&$value, &$key){

$value = strtoupper($value);

}

array\_walk\_recursive($combine, "setCase");

**echo** "<pre>";

var\_dump($combine);

**echo** "</pre>";

?>

**Code 13: passiveiteration.php**

#### Sorting Arrays

**sort($array)**

It modifies the actual array. This means that we cannot call this function by passing anything other than a single variable to it. It effectively destroys all the keys in the array and renumbers its elements starting from zero.

**asort($array)**

It is similar to sort but it maintains the key association

Both sort() and asort() accept a second optional parameter

|  |  |
| --- | --- |
| **SORT\_REGULAR** | Compare items as they appear in the array without conversion |
| **SORT\_NUMERIC** | Convert each element to a numeric value for sorting purposes |
| **SORT\_STRING** | Compare all elements in strings |

**rsort($array), arsort($array)**

Sorts the array in reverse order

**natsort($array)**

Natural sorting. The string value of “10t” will be considered “lower” than “2t” because it starts with the character 1, which has a lower value than 2.

**natcasesort($array)**

Natural case-insensitive version of natural sort

**ksort($array)**

Sort the array using the keys

**krsort($array)**

Sort the array using the keys in reverse order

**usort($array)**

User defined sorting of an array.

One of the custom defined array could be, to sort an arrat according to the length of each elements’s string representation. If the elements are of same length use string compare function strcmp($string, $string)

**uksort($array)**

User defined sorting of an array by key.

**ursort($array)**

User defined sorting of an array by key in reverse order.

**strcmp($string, $string)**

Compares the length of the two strings

#### Anti-Sort

**shuffle($array)**

Scramble the order of the array contents

**array\_keys($array)**

Saves the keys with its value

<?php

*//sort*

**echo** "sort<br/>";

$groceries = ["a"=>'apple', "o" =>'orange', "l" => 'lemon'];

sort($groceries);

*//It destroys the keys*

**echo** "<pre>";

var\_dump($groceries);

**echo** "</pre>";

*//asort*

**echo** "asort<br/>";

$groceries = ["a"=>'apple', "o" =>'orange', "l" => 'lemon'];

asort($groceries);

*//It maintains the keys*

**echo** "<pre>";

var\_dump($groceries);

**echo** "</pre>";

**echo** "===================================================";

**echo** "<br/>";

*//rsort*

**echo** "rsort<br/>";

$groceries = ["a"=>'apple', "o" =>'orange', "l" => 'lemon'];

rsort($groceries);

*//It destroys the keys*

**echo** "<pre>";

var\_dump($groceries);

**echo** "</pre>";

*//arsort*

**echo** "arsort<br/>";

$groceries = ["a"=>'apple', "o" =>'orange', "l" => 'lemon'];

arsort($groceries);

*//It maintains the keys*

**echo** "<pre>";

var\_dump($groceries);

**echo** "</pre>";

**echo** "===================================================";

**echo** "<br/>";

*//natsort*

**echo** "natsort<br/>";

$groceries = ["a"=>'10T', "O" =>'20t', "l" => '30t', "f" => '40T'];

natsort($groceries);

**echo** "<pre>";

var\_dump($groceries);

**echo** "</pre>";

*//natcasesort*

**echo** "natcasesort<br/>";

$groceries = ["a"=>'10T', "O" =>'20t', "l" => '30t', "f" => '40T'];

natcasesort($groceries);

**echo** "<pre>";

var\_dump($groceries);

**echo** "</pre>";

**echo** "===================================================";

**echo** "<br/>";

*//ksort*

**echo** "ksort<br/>";

$groceries = ["a"=>'10T', "O" =>'20t', "l" => '30t', "1" => '40T'];

ksort($groceries);

**echo** "<pre>";

var\_dump($groceries);

**echo** "</pre>";

*//krsort*

**echo** "krsort<br/>";

$groceries = ["a"=>'10T', "O" =>'20t', "l" => '30t', "1" => '40T'];

krsort($groceries);

**echo** "<pre>";

var\_dump($groceries);

**echo** "</pre>";

**echo** "===================================================";

**echo** "<br/>";

*//usort*

**echo** "usort<br/>";

$items = ["three", "2two", "one", "two"];

usort($items, "organize");

**function** organize($left, $right){

$diff = strlen($left) - strlen($right);

**if**(!$diff):

**return** strcmp($left, $right);

**endif**;

**return** $diff;

}

**echo** "<pre>";

var\_dump($items);

**echo** "</pre>";

**echo** "===================================================";

**echo** "<br/>";

*//shuffle*

**echo** "shuffle<br/>";

$items = ["a"=>'apple', "o" =>'orange', "l" => 'lemon'];

shuffle($items);

*//It destroys the keys*

**echo** "<pre>";

var\_dump($items);

**echo** "</pre>";

**echo** "===================================================";

**echo** "<br/>";

?>

**Code 14: sort.php**

#### Arrays as Stacks, Queues and Sets

Arrays are often used as **stacks** (**Last in, First Out, or LIFO**)

Array are also used as **queues** (**First in, First Out, or FIFO**)

PHP simplifies the approach by providing a set of functions that can be used to push an pull for stacks and shift and unshift for queues

*Add item at the end and remove the item at the end*

**array\_push($stack, “$item”, “item”)**

Add elements at the end of the array

**array\_pop($stack)**

Remove element at the end of the array

*Add item at the start and remove the item at the start*

**array\_shift($queues)**

Remove element at the start of the array

**array\_unshift($queues, “$item”)**

Add element at the start of the array

<?php

*/\*\**

*\* Stack (Last In, First Out)*

*\*/*

**echo** "Array Push";

$list = **array**("guitar", "amp", "cables");

array\_push($list, "keyboard", "drums");

**echo** "<pre>";

var\_dump($list);

**echo** "</pre>";

**echo** "============================";

**echo** "<br/>";

**echo** "Array Pop";

$last\_in = array\_pop($list);

**echo** "<pre>";

var\_dump($last\_in);

var\_dump($list);

**echo** "</pre>";

**echo** "============================";

**echo** "<br/>";

*/\*\**

*\* Queues (First in, First Out)*

*\*/*

**echo** "Array Shift";

$list = **array**("guitar", "amp", "cables");

array\_shift($list);

**echo** "<pre>";

var\_dump($list);

**echo** "</pre>";

**echo** "============================";

**echo** "<br/>";

**echo** "Array Unshift";

array\_unshift($list, "keyboard", "drums");

**echo** "<pre>";

var\_dump($last\_in);

var\_dump($list);

**echo** "</pre>";

**echo** "============================";

**echo** "<br/>";

?>

**Code 15: stackqueues.php**

#### Set Functionalities

**array\_diff($array, $array)**

It is used to compute the difference between two arrays

**array\_diff\_assoc($array, $array)**

It is used to compute the difference between two arrays based on key value pairs

**array\_diff\_key($array, $array)**

It is used to compute the difference between two keys

**array\_diff\_uassoc($array, $array)**

User define differences between two arrays based on key value pairs

**array\_diff\_ukey($array, $array)**

User define difference between two keys

**array\_intersect($array, $array)**

Common section of two arrays

**array\_intersect\_key($array, $array)**

Common keys between two arrays

**array\_intersect\_assoc($array, $array)**

Common section between two arrays using key value pairs

**array\_intersect\_ukey($array, $array)**

User define common section of two array using key

**array\_intersect\_uassoc($array, $array)**

User define common section of two array using key value pair

<?php

**echo** "array diff<br/>";

$collection = [299, 23, 15, 90];

$collection2 = [23, 15, 100, 70];

**echo** "<pre>";

var\_dump(array\_diff($collection, $collection2));

**echo** "</pre>";

**echo** "=====================================<br/>";

**echo** "array intersect<br/>";

$collection = [299, 23, 15, 90];

$collection2 = [23, 15, 100, 70];

**echo** "<pre>";

var\_dump(array\_intersect($collection, $collection2));

**echo** "</pre>";

?>

**Code 16: setfunctionalities.php**

### Strings and Patterns

Escape sequences are sometimes called control characters and take the form of a backslash followed by one or more characters.

Encapsulate the variables in curly braces.

**Heredoc Syntax**

It can be used to declare complex strings.

**Binary safe**: This means that all characters in the string are counted, reagardless of their value. In some languages notably C, some functions are designed to work with “zero-terminated” strings, where was the NULL character is used to signal the end of the string.

<?php

*/\*\**

*\* Encapsulation*

*\*/*

**echo** "Encapsulation</br>";

$describe = "toy";

$persons = ["John", "Dwayne", "Mary"];

**echo** "There are loads of **{**$describe**}**s in the store"."<br/>";

**echo** "Citation: **{**$persons[1]**}**[1984]<br/>";

**echo** nl2br("**\n**");

*/\*\**

*\* Herodoc*

*\*/*

**echo** "Herodoc</br>";

$describe = "toy";

**echo** <<<BOOM

There are loads of {$describe}s in the store.

BOOM;

**echo** nl2br("**\n**");

?>

**Code 17: encapsulation.php**

#### Compare string

**strcmp($string, $string)**

Check whether the string matches or not, sensitive to cases.

**strcasecmp($string, $string), substr\_compare($string, $string)**

Check whether the string matches or not, insensitive to cases.

**strpos($haystack, $string)**

It allows us to find the position of substring(***needle***), inside a string (***haystack***)

**strstr($string, $word)**

Searches for the first occurrence of a string inside another string. Then reveals everything after that.

**stripos()**

It allows us to find the position of substring(***needle***), inside a string (***haystack***) (case-insensitive)

**stristr($string, $word)**

Searches for the first occurrence of a string inside another string. Then reveals everything after that. (case-insensitive)

**strspn($string, $mask, $starting\_position, $ending\_position)**

Match a string against a “whitelist” mask of allowed characteristics. It returns the length of the intial segment of the string that contains any of the characters specified in the mask.

**strcspn($string, $mask, $starting\_position, $ending\_position)**

Match a string against a “blacklist” mask of allowed characteristics.

<?php

*/\*\**

*\* Shows find the letter prints it after that searched letter*

*\*/*

**echo** "strstr<br/>";

$string = "abc";

**echo** strstr($string, "b")."<br/>";

**echo** "==============";

**echo** "<br/>";

**echo** "String Array<br/>";

$string = "abc";

**echo** $string[1]."<br/>";

**echo** "==============";

**echo** "<br/>";

*/\*\**

*\* Compares between two strings (Case Insensitive)*

*\*/*

**echo** "String Compare<br/>";

$string = "I am a good boy";

**echo** "<pre>";

var\_dump(strcasecmp($string, "i am a good boy"));

**echo** "</pre>";

**echo** "==============";

**echo** "<br/>";

*/\*\**

*\* WhiteList*

*\*/*

**echo** "strspn<br/>";

$haystack = "aaaabbbccddasdf";

$needle = "abcd";

**echo** strspn($haystack, $needle)."<br/>";

**echo** "==============";

**echo** "<br/>";

?>

**Code 18: transformer.php**

#### Simple Search and Replace Operations

**str\_replace($substring, $replace, $string)**

Replacing portions of a string with a different substring.

**str\_ireplace($substring, $replace, $string)**

Replacing portions of a string with a different substring. Case in-sensitive

**substr\_replace($string, $replace, $start\_location, $how\_many)**

Replacing portions of a string with a different substring.

**substr($string, $start\_location, $end\_location)**

Portion of a string

<?php

*//String Replace*

**echo** "<strong> String Replace </strong></br>";

**echo** str\_replace('Vroom', 'Boom','Vroom Shakalaka Vroom Shakalaka Vroom',$a).", **$a** characters got replaced"."<br/>";

**echo** str\_ireplace('vroom', 'Boom','Vroom Shakalaka Vroom Shakalaka Vroom',$b).", **$b** characters got replaced"."<br/>";

*//Passing Array*

**echo** str\_replace(['Vroom', 'Shakalaka'], ['Boom', 'Shakalaka'],'Vroom Shakalaka Vroom Shakalaka Vroom')."<br/>";

**echo** str\_ireplace(['vroom', 'shakalaka'], ['Boom', 'Shakalaka'],'Vroom Shakalaka Vroom Shakalaka Vroom')."<br/>";

**echo** str\_ireplace(['vroom', 'shakalaka'], 'Boom','Vroom Shakalaka Vroom Shakalaka Vroom')."<br/>";

**echo** "</br>";

**echo** "<strong> Substring Replace </strong></br>";

**echo** substr\_replace('Hello World', "Reader", 6);

**echo** substr\_replace('I am a disco dancer', "bouncer", 13, 6);

**echo** "</br>";

**echo** "</br>";

*//Combining strpos + substr\_replace*

**echo** "<strong> Combining strpos + substr\_replace </strong></br>";

$user = "khan.photon@gmail.com";

$name = substr\_replace($user, " ", strpos($user, "@"));

**echo** $name;

*//Extracting Substring*

$name = "James Rozario";

**echo** substr($name, 0, 5);

?>

**Code 19: stringreplace.php**

#### Formatting Strings

**printf($string)**

It is used to print character stream of data on stdout console.

**sprintf($string, $datatype)**

It writes a formatted string to a variable.

**fprintf($file\_path, $string, $datatype)**

It writes a formatted string to a variable.

<?php

*/\*\**

*\* Works on Linux*

*\*/*

*// echo setlocale(LC\_ALL, "ja\_JP");*

*// echo money\_format("%i", "1000000.698");*

*/\*\**

*\* printf*

*\*/*

printf("Returns a string with a return value of 1");

**echo** "<br/>";

*/\*\**

*\* sprintf*

*\*/*

$number = 1;

$word = "formatted";

$txt = sprintf("Writes %s string to %i or more variables", $word, $number);

**echo** $txt;

**echo** "<br/>";

*/\*\**

*\* fprintf*

*\*/*

$number = 1;

$word = "formatted";

**try**{

$file = fopen("sample.txt", "w");

fprintf($file, "Writes %s string to %u or more variables", $word, $number);

}

**catch**(except $e){

"Error: ".$e->getMessage();

}

**finally**{

fclose($file);

}

?>

*/\*\**

*\* sscanf*

*\*/*

$data = "123 456 789";

$format = "%d %d %d";

**echo** "<pre>";

var\_dump(sscanf($data, $format));

**echo** "</pre>";

**Code 20:formatting.php**

### Web Programming

**urlencode($string)**

This function is convenient when encoding a string to be used in a query part of a URL, as a convenient way to pass variables to the next page.

<!doctype html>

<html>

<head>

<title> Alternative Form </title>

<meta charset = "UTF-8">

</head>

<body>

<form method = "get" action = "<?php echo **$\_SERVER['PHP\_SELF']**; ?>">

<label **for** = "fullname"> Full Name </label><br/>

<input type = "text" name = "fullname"/><br/><br/>

<label **for** = "gender"> Gender </label><br/>

Male <input type = "radio" name = "gender" value = "male">

Female<input type = "radio" name = "gender" value = "female"><br/><br/>

<label **for** = "car"> Car </label><br/>

<select name = "car">

<option value = "Toyota">Toyota</option>

<option value = "Nissan">Nissan</option>

<option value = "Tata">Maxda</option>

</select><br/><br/>

<label **for** = "pay"> Pay </label><br/>

<input type = "checkbox" name = "pay" value = "paid"> Paid

<input type = "checkbox" name = "pay" value = "notpaid"> **Not** Paid<br/><br/>

<input type = "submit" value = "Confirm" name = "submit"/>

</form><br/>

</body>

</html>

**Code 21: leftovers.php**

If we need to write a script that supposed to work just as well with both GET and POST requests, we can use **$\_REQUEST** superglobal array. **$\_REQUEST** can contain **GET, POST and cookie information**

If a user wants to upload a file we need to check check **$\_FILE[‘error’], which should be zero, Check the filesize and tmp\_name should not be set to none.**

**is\_uploaded\_file($file)**

Check whether the file is uploaded or not

#### HTTP Headers

##### Compression

**ob\_start(“ob\_hzhandler”)**

HTTP supports the transparent compression and decompression of data in transit during a transaction using the gzip algorithm. Compression will make a considerable impact on badwidth usage-as much as a 90% decrease in file size. It is performed on the fly, it uses up many more resourced than a typical request.

##### Caching

**header(“Cache-Control: no-cache, must-revalidate”)**

**header(“Expires: Thu,31 May 2002 04:34:00 GMT”)**

This set of headers tells the browser not to cache the item at all by setting a cache expiration date in the past.

### Object Oriented Programming(OOP)

It revolves around the concept of grouping code and data together in logical units called **classes**. The process is usually referred to as **encapsulation, or information hiding.** Classes are representation of a set of functions (also called **methods**) and variables (**called properties**) designed to work together and to provide a specific interface to the outside world. Classes are **blueprints** that cannot be used directly, they must be instanstiated into objects.

<?php

*/\*\**

*\* Object Oriented Programming*

*\*/*

**class** **human**{

**protected** $name;

**const** BAR = "Hello World";

**public** **function** \_\_construct($name){

$this->name = $name;

}

**public** **function** move(){

**return** $this->name.", I move and move!";

}

**public** **function** text(){

**return** human::BAR;

}

}

$peter = **new** human("Peter Parker");

**echo** $peter->move()."<br/>";

**echo** $peter->text();

?>

**Code 22: oop.php**

##### Inheritance

This allows a class to extend another class, essentially adding new methods and properties as well as overriding existing ones as needed.

<?php

**class** **parents**{

**public** **function** duty(){

**echo** "Parents: I work all day!<br/>";

}

}

**class** **child** **extends** parents{

**public** **function** duty(){

**echo** "Child: I party all day<br/>";

}

**public** **function** mom(){

**parent**::duty();

}

}

**class** **uncle** **extends** child{

**public** **function** duty(){

parents::duty();

}

}

**class** **grandchild** **extends** child{

**public** **function** duty(){

**echo** "Grandchild: I sleep all day<br/>";

}

}

$parents = **new** parents();

$parents->duty();

$child = **new** child();

$child->duty();

$child->mom();

$uncle = **new** uncle();

$uncle->duty();

?>

**Code 23: inheritance.php**

##### Visibility

|  |  |
| --- | --- |
| **Public** | The resource be accessed from any scope |
| **Protected** | It can only be accessed within a class and its descendants |
| **Private** | It can only be accessed within a class |
| **Final** | It can be accessed from any scope but it cannot be changed |

\*final 🡪 It only applies to method and classes

**get\_object\_vars($object)**

Gets the accessible non-static properties of the given object according to scope.

##### Interfaces and Abstract Classes

An abstract class essentially defines the basic skeleton of a specific type of encapsulated entity.

Therefore, in this type of class one of the method needs to be abstract. It means when a class is extended, it should have one of its method from the parent class. This class cannot be instantiated.

<?php

**abstract** **class** **car**{

**protected** $brand;

**protected** $wheels;

**protected** $doors;

**abstract** **protected** **function** wheels($wheels);

**abstract** **protected** **function** doors($doors);

**public** **function** describe(){

**echo** "**$this->brand**: It has **$this->wheels** wheels and **$this->doors** doors";

}

}

**class** **toyota** **extends** car{

**public** **function** \_\_construct($brand){

$this->brand = $brand;

}

**public** **function** wheels($wheels){

$this->wheels = $wheels;

}

**public** **function** doors($doors){

$this->doors = $doors;

}

}

$toyota = **new** toyota("Corolla");

$toyota->wheels(4);

$toyota->doors(4);

$toyota->describe();

?>

**Code 24: abstract.php**

**Interfaces**

It is used to specify an API (Application Programming Interface) that a class must implement. This allows us to create a common contract that our classes must implement in order to satisfy certain logical requirements. A class can only extend one parent class, but interface can be implement multiple interfaces.

<?php

**interface** dress{

**public** **function** stitches();

**public** **function** material();

**public** **function** dye();

**public** **function** lining();

}

**interface** shop{

**public** **function** tailor();

}

**class** **salwarkameez** **implements** dress, shop{

**public** **function** stitches(){

**return** "stitch";

}

**public** **function** material(){

**return** "botton";

}

**public** **function** dye(){

**return** "blue";

}

**public** **function** lining(){

**return** "lining";

}

**public** **function** tailor(){

**return** 25;

}

}

$salwarkameez = **new** salwarkameez();

**echo** $salwarkameez->stitches(). $salwarkameez->material().$salwarkameez->dye().$salwarkameez->lining().$salwarkameez->tailor();

?>

**Code 25: interface.php**

#### Throwing Exceptions

Exeptions are usually created and thrown when an error occurs by using the throw construct

<?php

**try**{

**throw** **new** myException("What is this problem!");

}**catch**(myException $e){

**echo** "Error: ".$e->getMessage();

}

**class** **myException** **extends** Exception{

}

?>

**Code 26: exception.php**

#### Lazy Loading

<?php

spl\_autoload\_register(**function**($class){

**require\_once**($class.".php");

});

$helper = **new** helper();

$helper = **new** helpertwo();

?>

<?php

**class** **helper**{

**public** **function** \_\_construct(){

**echo** "helper";

}

}

?>

<?php

**class** **helpertwo**{

**public** **function** \_\_construct(){

**echo** "helpertwo";

}

}

?>

**Code 27: lazyload.php**

**spl\_autoload\_register(function($class){**

**require\_once($class.".php");**

**});**

Includes all the classes by itself

#### Reflection

Reflection API is a collection of functions and objects that allows us to examine the contents of a scripts’s code, such as functions objects, at runtime. Reflections can be very handy in a number of circumstances; it can be used to generate simple documentation, or for determining whether certain functionality is available to a script, and so on. If we want to expand, we can simple use ReflectionClass and ReflectionMethod

<?php

*/\*\**

*\* Creating class called Hello*

*\* which return text*

*\*/*

**class** **hello**{

*/\*\**

*\* $text takes the data from the user input*

*\* @var string*

*\*/*

**protected** $text;

*/\*\**

*\* \_\_construct takes the string from the user*

*\* @param string $text*

*\*/*

**public** **function** \_\_construct($text){

$this->text = $text;

}

*/\*\**

*\* getText return the text*

*\* @return string*

*\*/*

**public** **function** getText(){

**return** $text;

}

}

$reflector = **new** ReflectionClass('hello');

**echo** "<pre>";

var\_dump($reflector->getDocComment());

var\_dump($reflector->getName());

var\_dump($reflector->getMethods());

var\_dump($reflector->getStartLine());

var\_dump($reflector->getEndLine());

**echo** "</pre>";

?>

**Code 27: reflection.php**

### Design Pattern

#### Singleton Pattern

Its goal is to provide access to a single resource that is never duplicated, but that is made available to any portion of an application that requests it without the need to keep track of its existence.

Static properties cannot be accessed through the object using the arrow operator 🡪

*/\*\**

*\* Singleton Pattern*

*\*/*

**class** **database**{

**private** **static** $\_db\_con;

**private** $\_connection;

**private** $server\_name = "localhost";

**private** $user\_name = "root";

**private** $password = "";

**private** $db\_name = "riskyjobs";

**private** **function** \_\_construct(){

$this->\_connection = mysqli\_connect($this->server\_name, $this->user\_name, $this->password, $this->db\_name)

**or** **die**("Server Connection Denied");

}

**public** **function** connect(){

**if**(is\_null(self::$\_db\_con)):

self::$\_db\_con = **new** database();

**endif**;

**return** self::$\_db\_con;

}

}

$database = database::connect();

?>

**Code 28: singleton.php**

Things to point out: The constructor is set to **private**, therefore the class cannot be instanstiated, the variable(properties) singleton on is static. The **static property** can be accessed be **self∷$property**

or by **class\_name∷$property** the method of the class can be constructed using **class\_name∷method\_name().** **Constants** are accessed in the similar way. The most typical example of this pattern is a datavase connection, which normally only needs to be created once at the beginning of a script and then used throughout its code.

#### Factory Pattern

It is used in scenarious where we have a generic class (the factory\_ that provides the facilities for creating instances of one or more separate “specialized” classes that handle the same task in different ways.

It provides an excellent solution in the management of multiple storage mechanisms for a given task.

|  |  |
| --- | --- |
| **Static** | **Constants** |
| Their values can be changed | Theie values cannot be changed |
| They can be protected, private or public | They are always public |
| They both can be accessed using Class∷$property | |

**class** **Configuration**{

**const** STORE\_INI = 1;

**const** STORE\_DB = 2;

**const** STORE\_XML = 3;

**public** **static** **function** getStore($type = self::STORE\_XML){

**switch**($type):

**case** self::STORE\_INI:

**return** **new** Configuration\_INI();

**break**;

**case** self::STORE\_DB:

**return** **new** Configuration\_DB();

**break**;

**case** self::STORE\_XML:

**return** **new** Configuration\_XML();

**break**;

**default**:

**throw** **new** Exception("Unkown Datastore Specified");

**endswitch**;

}

}

**class** **Configuration\_INI**{

**public** **function** \_\_construct(){

**echo** "Storing in Configuration\_INI";

}

}

**class** **Configuration\_DB**{

**public** **function** \_\_construct(){

**echo** "Storing in Configuration\_DB";

}

}

**class** **Configuration\_XML**{

**public** **function** \_\_construct(){

**echo** "Storing in Configuration\_XML";

}

}

$configuration = Configuration::getStore(Configuration::STORE\_DB);

$reflect = **new** ReflectionClass($configuration);

**echo** "<pre>";

var\_dump($reflect->getName());

var\_dump($reflect->getMethods());

**echo** "</pre>";

?>

**Code 29: factory.php**

#### Registry Pattern

This is done by taking the **singleton pattern** little further. This allows us to use any object as a Singleton without it being written specifically that way. It is most used when we need t connect to an alternate database to perform a small set of tasks every now and then.

**is\_null($object)**

Checks whether the object is null or not

**is\_object($object)**

Checks whether the variable is an object or not

**get\_class($object)**

Returns the name of the class of an object

<?php

**class** **db**{}

**class** **db2**{}

**class** **Registry**{

**private** **static** $register;

**public** **static** **function** add(&$item){

$name = get\_class($item);

self::$register[$name] = $item;

}

**public** **static** **function** &get($name){

**if**(array\_key\_exists($name, self::$register)):

**return** self::$register[$name];

**else**:

$msg = "**$name** is not registered";

**throw** **new** Exception($msg);

**endif**;

}

**public** **static** **function** exist($name){

**if**(array\_key\_exists($name, self::$register)):

**return** **true**;

**else**:

**return** **false**;

**endif**;

}

}

$db = **new** db();

$db2 = **new** db2();

Registry::add($db);

**if**(Registry::exist('db')):

**echo** "<pre>";

$db = Registry::get('db');

$reflector = **new** ReflectionClass($db);

var\_dump($reflector->getName());

var\_dump($reflector->getMethods());

**echo** "</pre>";

**endif**

?>

**Code 30: registry.php**

#### Standard PHP Librarary

The Standard PHP Library (SPL) allow user to loop through the objects.

##### Access Objects as Arrays

**offsetSet()**

It sets a value in the array

**offsetGet()**

retrieves from the array

**offsetUnset()**

removes a value from the array

**offsetExists()**

It sets a value in the array

Therefore, the built-in interface contains

**interface** ArrayAccess{

**abstract** **public** **function** offsetSet($offset);

**abstract** **public** **function** offsetGet($offset);

**abstract** **public** **function** offsetUnset($offset);

**abstract** **public** **function** offsetExists($offset);

}

<?php

**class** **first**{};

**class** **second**{};

**class** **myArray** **implements** ArrayAccess{

**protected** $container = **array**();

**public** **function** \_\_construct($offsets){

**if**(is\_array($offsets)):

**foreach**($offsets **as** $offset):

$name = get\_class($offset);

$this->offsetSet($offset, $name);

**endforeach**;

**else**:

$this->offsetSet($offsets);

**endif**;

}

**public** **function** offsetSet($offset, $name){

$this->container[$name] = $offset;

}

**public** **function** offsetGet($offset){

**return** $this->container[get\_class($offset)];

}

**public** **function** offsetUnset($offset){

**if**(array\_key\_exists(get\_class($offset), $this->container)):

unset($this->container[get\_class($offset)]);

**else**:

$msg = getclass($offset).", object do not exist!";

**throw** **new** Exception($msg);

**endif**;

}

**public** **function** offsetExists($offset){

**return** array\_key\_exists(get\_class($offset), $this->container);

}

}

$first = **new** first();

$second = **new** second();

$array = **new** myArray([$first,$second]);

**echo** "<pre>";

**echo** "<strong>Array of Objects</strong><br/><br/>";

var\_dump($array);

**echo** "<strong>Get the the Specific Object</strong><br/><br/>";

var\_dump($array->offsetGet($first));

**echo** "<strong>Delete one of the Object</strong><br/><br/>";

$array->offsetUnset($second);

*// $array->offsetUnset($third);*

var\_dump($array);

**echo** "<strong>Check whether the Object exists or not</strong><br/><br/>";

var\_dump($array->offsetExists($first));

var\_dump($array->offsetExists($second));

**echo** "</pre>";

?>

**Code 31: arrayaccess.php**

##### Simple Iterators

The Iterator interface is the simplest of the iterator family, providing simple iteration over any single-dimension array.

The interface looks like

**interface** Iterator{

**public** **abstract** **function** current();

**public** **abstract** **function** next();

**public** **abstract** **function** rewind();

**public** **abstract** **function** key();

**public** **abstract** **function** valid();

}

<?php

**class** **myIterator** **implements** Iterator{

**protected** $data = **array**();

**protected** $location = 0;

**public** **function** \_\_construct($datas){

**foreach**($datas **as** $data):

$this->data[] = $data;

**endforeach**;

}

**public** **function** current(){

**return** $this->data[$this->location];

}

**public** **function** next(){

$this->location += 1;

}

**public** **function** rewind(){

$this->location = 0;

}

**public** **function** key(){

**return** $this->location;

}

**public** **function** valid(){

**return** isset($this->data[$this->location]);

}

}

**class** **first**{}

**class** **second**{}

$first = **new** first();

$second = **new** second();

$third = **new** first();

$fourth = **new** second();

$fifth = **new** first();

$collection = [$first, $second, $third, $fourth, $fifth];

**echo** "<pre>";

**echo** "<strong>All Objects in the Array</strong><br/>";

$data = **new** myIterator($collection);

var\_dump($data);

**echo** "<br/><strong>Next Array</strong><br/>";

$data->next();

$data->next();

var\_dump($data->key());

var\_dump(get\_class($data->current()));

$data->rewind();

var\_dump($data->key());

**echo** "</pre>";

?>

**Code 32: iterator.php**

##### Seekable Iterators

It extends the iterator interface and adds a **seek()**method enables the ability to retrieve a specific item from internal data store.

**interface** SeekableIterator{

**public** **abstract** **function** current();

**public** **abstract** **function** next();

**public** **abstract** **function** rewind();

**public** **abstract** **function** key();

**public** **abstract** **function** valid();

**public** **abstract** **function** seek($index);

}

**Code 33: seekableiterator.php**

##### Recursive Iterators

It allows looping over multi-dimensional tree-like data structures. SimpleXML, uses recursive tieration to allow looping through complex XML document trees. It is not an interface.

**str\_repeat($string, $repeat)**

It repeats a string a specified number of times.

Recursion occurs when something contains, or uses, a similar version of itself. Similar version that contains or uses another similar version itself, and so on. Recursion can also refer to a method of problem solving that first solves a smaller version of the problem and then uses that result plus some other computation to formulate an answer to the original problem.

<?php

$companies = **array**(

**array**("Acme Anvil Co."),

**array**(

**array**(

"Human Resources",**array**(

"Tom","Dick","Harry")),

**array**(

"Accounting", **array**(

"Zoe", "Duncan", "Jack", "Jane")

)

)

);

$arrayiter = **new** RecursiveArrayIterator($companies);

$iteriter = **new** RecursiveIteratorIterator($arrayiter);

**foreach** ($iteriter **as** $key => $value){

$depth = $iteriter->getDepth();

**switch**($depth):

**case** 1:

**echo** "<h1>**$value**</h1>";

**break**;

**case** 2:

**echo** "<h2>**$value**</h2>";

**break**;

**case** 3:

**echo** "<li>**$value**</li>";

**break**;

**default**:

**endswitch**;

}

?>

**Code 34: recursiveiteratoriterator.php**

##### Filtering Iterators

It can be used to filter the items returned by an iteration. The accept() method simply determines whether any given element should be allowed in the iteration.

<?php

**class** **myFilterIterator** **extends** FilterIterator{

**private** $user\_filter;

**public** **function** \_\_construct($iterator, $user\_filter){

**parent**::\_\_construct($iterator);

$this->user\_filter = $user\_filter;

}

**public** **function** accept(){

$user = $this->getInnerIterator()->current();

**if**(strcasecmp($user['name'],$this->user\_filter) == 0):

**return** **false**;

**endif**;

**return** **true**;

}

}

$array = **array**(

**array**('name' => 'jonathan','id' => '5'),

**array**('name' => 'abdul' ,'id' => '22')

);

$object = **new** ArrayIterator($array);

$iterator = **new** myFilterIterator($object,'abdul');

**foreach** ($iterator **as** $result) {

**echo** $result['name'];

}

?>

**Code 36: filteriterator.php**

##### Brief

**Singleton(Design Pattern)**

* Static property
* Private \_\_constructor
* Public method

🡪 static property = class

Class∷method()

#Single Resource Method

**Factory(Design Pattern)**

* Static method

🡪 Which class to you want to use

Single class to identify multiple class

**Registry(Design Pattern)**

* 3 static methods

Check + Add Objects + exist

**ArrayAccess(Interface)**

🡪set

🡪get

🡪unset

🡪exist

**Iterator(Interface)**

🡪current

🡪next

🡪rewind

🡪key

🡪valid

**RecursiveArrayIterator 🡪 Generic Class**

**RecursiveIteratorIterator 🡪 Generic Class**

**FilterIteratory 🡪 Abstract Class**

### XML and Web Services

**Extensible Markup Language** (XML) used for **Really Simple Syndication** and **Atom Feed Formats.** Web Services provide a way by which any computer may exchange data with another using the web as a transport medium. XML is a subset of Standard Generalized Markup Language (SGML).

#### Terminology

**Entity:** It is the unit of storage. (almost act like a variables)

**Element:** A data object that is part of an XML document. Elements may contain other elements or raw textual data, as well as feature zero or more attributes.

**Document Type Declataration:** A set of instruction sthat describes the accepted structure and content of an XML file. Like entities, DTDS can either be externally defined or embedded.

**Well-formed:** An XML document is considered well-formed when it contains single root elevel element, all tags are opened and closed properly

**Valid:** An XML is valid when it is both well-formed and obeys a referenced DTD.

<?xml version="1.0"?>

<!DOCTYPE message SYSTEM "message.dtd">

**<message>** Referencing a document type declaration **</message>**

**Code 37: simple.xml**

Unless we are working with a document type declaration (DTD) or XML Schema Definition (XSD), which provides an alternate method to describe a document, crating XML is a free-form process*.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **Author** | **Publisher** | **ISBN** |
| The Moon is a Harsh Mistress | R.A Heinlein | Orb | 0312863551 |
| Fahrenheit 451 | R. Bradbury | Del Rey | 0345342868 |
| The Silmarillion | J.R.R. Tolien | G Allen & Unwin | 0048231398 |
| 1984 | G Orwell | Signet | 04515294934 |
| Frankenstein | M. Shelley | Bedford | 031219126X |

**Elements**: book, title, author, and publisher.

The **sole attribute** of **the book element is isbn**

Therefore, the tags are the **elements which contains data**

The **attributes contain metadata**.

#### Parsing XML Document

All XML parsing is done by SimpleXML internally using Document Object Model (DOM) parsing model.

**simplexml\_load\_string()**

Loads an XML document from a string

**simplexml\_load\_file()**

Loads an XML document from a path

**file\_get\_contents()**

The reads a file into a string.

**new SimpleXMLElement($path, NULL, true)**

Object oriented way of parsing a XML file. The **second argumen**t optionally allows the ability to specify additional **libxml parameters** that influence the way the library parses the XML. The **third parameter**, is important because it **informs the constructor that the first argument represents the path to a file**, rather than a string that contains the XML data itself

The drawback to this approach is that it is necessary to know the names of every element and attribute in the XML document.

Instead we can use the **SimpleXMLElement∷children(), SimpleXMLElement∷attributes(),** as well as **SimpleXMLElement∷getName()**

**<library>**

**<book** isbn = "0312863551"**>**

**<title>**The Moon is a Harsh Mistress**</title>**

**<author>**R.A Heinlein**</author>**

**<publisher>**Orb**</publisher>**

**</book>**

**<book** isbn = "0345342868"**>**

**<title>**Fahrenheit 451**</title>**

**<author>**R. Bradbury**</author>**

**<publisher>**Del Rey**</publisher>**

**</book>**

**<book** isbn = "0048231398"**>**

**<title>**The Silmarillion**</title>**

**<author>**J.R.R. Tolien**</author>**

**<publisher>**G Allen **&amp;** Unwin**</publisher>**

**</book>**

**<book** isbn = "04515294934"**>**

**<title>**1984**</title>**

**<author>**G Orwell**</author>**

**<publisher>**Signet**</publisher>**

**</book>**

**<book** isbn = "031219126X"**>**

**<title>**Frankenstein**</title>**

**<author>**M. Shelley**</author>**

**<publisher>**Bedford**</publisher>**

**</book>**

**</library>**

<?php

*//Load an XML String (Procedural)*

$xmlstr = file\_get\_contents("library.xml");

$library\_string = simplexml\_load\_string($xmlstr);

*//Load an XML File (Procedural)*

$library\_procedural = simplexml\_load\_file("library.xml");

*//Load an XML file (Object)*

$books = **new** simpleXMLElement("library.xml", **NULL**, **true**);

*/\*\**

*\* Preferred way to parse*

*\*/*

**foreach**($books->children() **as** $book):

**echo** nl2br($book->getName()." ");

**foreach**($book->attributes() **as** $isbn):

**echo** nl2br($isbn->getName()." : **$isbn\n**");

**endforeach**;

**foreach**($book->children() **as** $info):

**echo** nl2br($info->getName()." : **$info\n**");

**endforeach**;

**echo** "<br/>";

**endforeach**;

**echo** "=========================================================<br/><br/>";

*/\*\**

*\* In this method we need to know the whole structure of the XML Element*

*\*/*

**foreach**($books **as** $book):

**echo** nl2br($book['isbn']."**\n**");

**echo** nl2br($book->title."**\n**");

**echo** nl2br($book->author."**\n**");

**echo** nl2br($book->publisher."**\n\n**");

**endforeach**;

?>

**Code 36: load.php**

#### XPath Queries

**XML path Language (XPath)** is used to access and search XML documents. It is used in extensively in **Extensible Stylesheet Language Transformations (XSLT)** and form ths basis of **XML Query (XQuery)**

**SimpleXMLElement∷xpath()**

?php

$books = **new** SimpleXMLElement("library.xml", **NULL**, **true**);

$xpaths = $books->xpath('/library/book/title');

*//Search the title elements*

**foreach**($xpaths **as** $xpath):

**echo** nl2br($xpath."**\n**");

**endforeach**;

*//Search for the first child element*

$result = $books->book[0]->xpath('title');

**echo** "<pre>";

var\_dump($result);

**echo** "</pre>";

**foreach**($result **as** $title):

**echo** $title."<br/>";

**endforeach**;

?>

**Code 37: xpath.php**

#### Modifying XML Elements

**SimpleXMLElement∷addChild()**

It accepts three parameters, the first of which is the name of the new element. The second is an optional value for this element, and the third is an optional namespace to child the child belongs.

**SimpleXMLElement∷addAttribute()**

<?php

$books = **new** SimpleXMLElement("library.xml", **NULL**, **true**);

$book = $books->addChild('book');

$book->addAttribute("isbn", "0812550706");

$book->addChild('title', "Ender's Game");

$book->addChild('author', "Orson Scott Card");

$book->addChild('publisher', "Tor Science Fiction");

header('Content-Type: text/xml');

**echo** $books->asXML();

?>

**Code 38: modify.php**

To delete element in XML we need to assign the element to NULL

<?php

$books = **new** SimpleXMLElement("library.xml", **NULL**, **true**);

$books->book[3] = **NULL**;

header('Content-Type: text/xml');

**echo** $books->asXML();

?>

**Code 38: delete.php**

#### Working with Namespaces

The use of XML namespaces allows a provider to associate certain element and attribute names with namespaces identified by URIs.

**SimpleXMLElement∷getDocNamespaces()**

It returns an array of all namespaces declared in the document.

**SimpleXMLElement∷getNamespaces(bool)**

It returns an array of the namespaces declared in the document.

<?xml version="1.0"?>

**<books** xmlns="https://sphotonkhan.com"

xmlns:menu="https://sphotonkhan.com/menu"

xmlns:pub="https://sphotonkhan.com/pub"

xmlns:contactus="https://sphotonkhan.com/contactus"**>**

**<book** isbn = "0312863551"**>**

**<title>**The Moon is a Harsh Mistress**</title>**

**<author>**R.A Heinlein**</author>**

**<pub:publisher>**Orb**</pub:publisher>**

**</book>**

**<book** isbn = "0345342868"**>**

**<title>**Fahrenheit 451**</title>**

**<author>**R. Bradbury**</author>**

**<pub:publisher>**Del Rey**</pub:publisher>**

**</book>**

**<book** isbn = "0048231398"**>**

**<title>**The Silmarillion**</title>**

**<author>**J.R.R. Tolien**</author>**

**<pub:publisher>**G Allen **&amp;** Unwin**</pub:publisher>**

**</book>**

**<book** isbn = "04515294934"**>**

**<title>**1984**</title>**

**<author>**G Orwell**</author>**

**<pub:publisher>**Signet**</pub:publisher>**

**</book>**

**<book** isbn = "031219126X"**>**

**<title>**Frankenstein**</title>**

**<author>**M. Shelley**</author>**

**<pub:publisher>**Bedford**</pub:publisher>**

**</book>**

**</books>**

**Code 39: books.xml**

<?php

*//Returns all the elements*

**echo** "<strong>Returns all the elements</strong><br/>";

$books = **new** SimpleXMLElement("books.xml", **NULL**, **true**);

$namespaces = $books->getDocNamespaces();

**foreach**($namespaces **as** $key => $value):

**echo** nl2br("**{**$key**}**: **{**$value**}\n**");

**endforeach**;

**echo** "<br/><br/>";

*//Returns elements with namespace in it*

**echo** "<strong>Returns elements with namespace in it</strong><br/>";

$namespaces = $books->getNamespaces(**true**);

**foreach**($namespaces **as** $key => $value):

**echo** nl2br("**{**$key**}**: **{**$value**}\n**");

**endforeach**;

?>

**Code 40: namespace.php**

#### Document Object Model (DOM)

It is similar like SimpleXMLElement, we can work over here with minimal effort

##### Loading and Saving XML Documents

There are two ways to import documents into a DOM tree, the first is by loading them from a file. Alternatively, we can load a document from a string – whish is handy when using REST Web services.

**DomDocument∷loadHtmlFile()**

Import HTML files

**DomDocument∷loadHTML()**

Import HTML strings

**DomDocument∷saveHTMLFile()**

Save HTML file

**DomDocument∷save()**

Save XML documents to a file

**DomDocument∷saveXML()**

Save XML documents to a string

<?php

*/\*\**

*\* First way to import xml document*

*\* Import documents into a DOM tree and then load them*

*\* @var DomDocument*

*\*/*

$domHTML = **new** DomDocument();

$domHTML->load("sampleHTML.html");

*/\*\**

*\* Second way to import xml document*

*\* Load the document using a string*

*\* This hand when using REST Web Services*

*\* @var DomDocument*

*\*/*

$file = file\_get\_contents("library.xml");

$domXML = **new** DomDocument();

$domXML->loadXML($file);

*/\*\**

*\* Save Files to HTML and XML*

*\*/*

$domHTML->saveHTMLFile("SampleCopy.html");

$domXML->save("bookstore.xml");

?>

**Code 41: domloadsave.php**

##### XPath Queries

XPath – It is more powerful than the SimpleXML equivalent

**//library/book**

The two forward slashes indicate library is the root element of the document and the single slash indicates book is a child.

**//library/book/author[text() = “R. Bradbury”]/..**

We can use **text()** in square braces to perform a comparison against the value of a node, and the trailing “/..” indicates the parent element (i.e move back up the tree one node)

**query()**

Performs Query

**evaluate()**

It is like query but it will return a number.

<?php

$dom = **new** DomDocument();

$dom-> load("library.xml");

$xpath = **new** DomXPath($dom);

$query = "//library/book/author";

$result = $xpath->query($query);

**foreach**($result **as** $data):

**echo** "<pre>";

**echo** $data->tagName.": ".$data->textContent."<br/>";

**echo** "</pre>";

**endforeach**;

$query = "//library/book/title";

$result = $xpath->query($query);

**foreach**($result **as** $data):

**echo** "<pre>";

**echo** $data->nodeName.": ".$data->nodeValue."<br/>";

**echo** "</pre>";

**endforeach**;

$query = "//library/book/title/text()";

$result = $xpath->query($query);

**foreach**($result **as** $data):

**echo** "<pre>";

**echo** $data->nodeValue."<br/>";

**echo** "</pre>";

**endforeach**;

?>

**Code 42: domxpath.php**

##### Modifying XML Documents

**DomDocument∷createElement();**

Create new element to a loaded document

**DomDocument∷createTextNode();**

Add new data to a loaded document

<?php

$dom = **new** DomDocument();

$dom->load("library.xml");

$book = $dom->createElement("book");

$book->setAttribute("isbn", "0973589825");

$title = $dom->createElement("title");

$text = $dom->createTextNode("PHP|architect's Guide to PHP Design Patterns");

$title->appendChild($text);

$book->appendChild($title);

$author = $dom->createElement("author", "Jason E. Sweat");

$book->appendChild($author);

$publisher = $dom->createElement("publisher", "Marco Tabini &amp; Associates, Inc.");

$book->appendchild($publisher);

$dom->documentElement->appendChild($book);

?>

**Code 43: dommodify.php**

**DomNode∷appendChild()**

**DomNode∷insertBefore()**

It will move the node to the new location

**DomNode∷cloneNode()**

It will duplicate the node.

**DomNode∷saveXML()**

It will save the XML file.

**DomNode∷removeAttribute()**

It will remove the attribute.

**DomNode∷removeChild()**

It will remove the child.

**DomCharacterDate∷()**

It will remove the child.

**DomDocument∷createElementNS()**

Creates namespace element

**DomNode∷setAttributeNS()**

Sets attribute to the namespace

**dom\_import\_simplexml()**

Use the xml as both **simplexml object** as well **DOM object**

**ucwords()**

It converts the first character of each word in a string to uppercase.

**htmlentities()**

It converts characters to HTML entities.

#### Web Services

It provides a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. There are three popular types of Web Services

XML-RC, SOAP (the successor to web service) and REST

##### SOAP

**Simple Object Access Protocol,** it is a powerful tool ofr communication between disparate systesm, as it allows the definition and exchange of complex data types in both the request and response. It provides a mechanism for **Remote Procedure Call (RPC).** It is defined by using a **Web Service Description Language (WSDL) document.**

**Remote Proceduce Call (RPC)**

It is a mechanism for various message patterns

**SoapClient Class**

##### Accessing SOAP-based Web Services

It provides a one-stop solution to creats a SOAP client we just need to provide with the path to a WSDL file, and it will automatically build a PHP-friendly interface so that we can call directly from our scripts.

##### Debugging

**Soap∷\_\_getLastReqeustHeaders()**

**Soap∷\_\_getLastRequest();**

This are the special methods to make it possible to ebug messages sent to and received from a SOAP server.

##### Creating SOAP-based Web Services

**SoapServer()**

**SoapClient()**

##### REST

**Representational State Transfer** is a Web service architectural style in which the focus is on the presence of resouces in the system. Each resource must be identified by global identifier-a **URI**

#### Brief

1. simplexml\_load\_string()
2. simplexml\_load\_file()
3. file\_get\_contents()
4. new SimpleXMLElement($path, NULL, true)
5. SimpleXMLElement∷children()
6. SimpleXMLElement∷attributes()
7. SimpleXMLElement∷getName()
8. SimpleXMLElement∷xpath()
9. SimpleXMLElement∷addChild()
10. SimpleXMLElement∷addAttribute()
11. SimpleXMLElement∷getDocNamespaces()
12. SimpleXMLElement∷getNamespaces(bool)
13. DomDocument∷loadHtmlFile()
14. DomDocument∷loadHTML()
15. DomDocument∷saveHTMLFile()
16. DomDocument∷save()
17. DomDocument∷saveXML()
18. DomXPath
19. query()
20. evaluate()
21. DomDocument∷createElement();
22. DomDocument∷createTextNode();
23. DomNode∷appendChild()
24. DomNode∷insertBefore()
25. DomNode∷cloneNode()
26. DomNode∷saveXML()
27. DomNode∷removeAttribute()
28. DomNode∷removeChild()
29. DomCharacterDate∷()
30. DomDocument∷createElementNS()
31. DomNode∷setAttributeNS()
32. dom\_import\_simplexml()
33. Soap∷\_\_getLastReqeustHeaders()
34. Soap∷\_\_getLastRequest();
35. SoapServer()
36. SoapClient()

### Security

It is best to assume all the data are tainted. Two most common approaches to filtering input are whitelist and blacklist filtering. The blacklist approach is the less restrivtive form of filtering that assumes the programmer knows everything should nto be allowed to pass through. Blacklists must be modified continually, and expanded as new attack vectors become apparent. On the other hand, whitelist filtering much more restrictive, yet it affords the programmer the ability to accept only the input he expects to receive

**ctype\_alpha($var)**

Check for alphabetic character(s)

**ctype\_alnum($var)**

Check for alphabetic character(s)

**htmlspecialchars()**

It converts some predefined characters to HTML entities.

The predefined characters are:

& (ampersand) 🡪 &amp;

" (double quote) 🡪 &quot;

' (single quote) 🡪 &#039;

< (less than) 🡪 &lt;

> (greater than) 🡪 &gt;

**Example:**

$str = "This is some <b>bold</b> text.";

echo htmlspecialchars($str);

result: This is some &lt;b&gt;bold&lt;/b&gt; text.

**htmlentities()**

It converts characters to HTML entities.

**Example**

<?php

$str = "<© W3Sçh°°¦§>";

echo htmlentities($str);

?>

result: &lt;&copy; W3S&ccedil;h&deg;&deg;&brvbar;&sect;&gt;

#### Website Security

It refers to the security of the elements of a website through which an attacker can interface with our application. These vulnerable points of entry include forms and URLs, which are the most likely and easiest candidates for a potential attack.

##### Spoofed Forms

A common method used by attackers is a sppofed form submission. There are various waysto spoof forms, the easiest of which is to simple copy a target form and execute it from a different location.

maxlength attribute restricts the length of content entered into the fields.

##### Cross-Site Scripting

Iti si one of the most common and best-known kinds of attacks. The simplicity of this attack and the number of vulnerable application since existence make it very attractive to malicious users. An XSS attack exploits the user’s trust in the application and is usually an effort to steal user information, such as cookies and other personally identifiable data. All applications that display input are at risk.

##### Cross-Site Request Forgeries

A cross-site request forgery(CSRF) is an attack that attempts to cause a victim to unknowingly send arbitrary HTTP requests, usually to URLs requiring privileged access and using the existing session of the victim to determine access.

##### Database Security

Tricking MySQL with comments. A double hyphen (--) is used in SQL to comment out the remainder of a line of SQL code. To make it work, it is double hyphen with a space (-- ), everything after it is ignored.

INSERT INTO inventory VALUES(0, NOW(), ‘Hacker’, ‘10000000’, ‘hacker.png’, 1) – hacker.png

1 is used to approve and everything after – is commented.

Form fields are security weak point for web applications because they allow users to enter data.

Dangerous characters are any characters that could possibly change the nature of any SQL-query, such as commas, quotes or – comment characters. Even the spaces at the end of a piece of data can prove harmful. SQL injections can be prevented by properly processing form data.

trim()

Eliminates Leading or trailing spaces

Ex: $name = trim($\_POST[‘name’]);

mysqli\_real\_escape\_string($connection, trim($\_POST[‘name]));

It converts dangerous characters into an escaped format that won’t adversely affect SQL queries.

##### Session Security

Two popular forms of session attacks are session fixation and session hijacking. The simple attack is called session fixation because the attacker fixes the session. This is most commonly achieved by creating a link to your application and appending the session identifier that the attacker wishes to give any user clicking the link. While the user accesses your site through this session, they may provide sensitive information or even login credentials. If theuser logs in while using the provided access to the user’s account. This is why session fixaton is sometimes referred to as “session riding.”

##### File System Security

PHP has the ability to directly access the filesystem and even execute shell commands. Remode Code Injection occurs when an attacker is able to cause our application to execute PHP code of their choosing (**include and require**)

##### Command Injection

PHP provides great power with **exec(), system(), passthr(),** as well as the ` backtick operator, It is important to take great care to ensure that attackers cannot inject and execute arbitrary system commands. PHP provides **escapeshellcmd()** and **escapeshellarg()** means to escape shell output

<?php

session\_start();

$token = md5(uniqid(rand(), **TRUE**));

$\_SESSION['token'] = $token;

$server = $\_SERVER['HTTP\_USER\_AGENT'];

?>

<!DOCTYPE html>

<html>

<head>

<title> Security Form </title>

<meta charset = "UTF-8"/>

</head>

<body>

<form method = "post" action = "process.php"/>

<input type = "hidden" name = "token" value = "<?php **echo** $token;?>"/>

<input type = "hidden" name = "server" value = "<?php **echo** $server;?>"/>

<label for = "username"> Username</label><br/>

<input text = "text" name = "username" maxlength = "30"/><br/><br/>

<label for = "password"> Password </label><br/>

<input text = "password" name = "password" maxlength = "15"><br/><br/>

<label for = "colors"> Colors </label>

<select name = "colors">

<option value = "red"> Red </option>

<option value = "blue"> Blue </option>

<option value = "green"> Green </option>

</select><br/><br/>

<input type = "submit" name = "submit" value = "Confirm"/>

</form>

</body>

</html>

**Code 44: form.php**

<?php

session\_start();

$error = **array**();

$clean = **array**();

**function** sanitize($data, $key){

**if**(!**empty**($data)):

**global** $error;

$data = htmlentities(htmlspecialchars($data));

**switch**($key):

**case** "username":

**return** (ctype\_alpha($data))? $data: "";

**break**;

**case** "password":

**return** (ctype\_alnum($data))? $data: "";

**break**;

**case** "colors":

**if**(in\_array($data, ["red","blue","green"])):

**return** $data;

**endif**;

**break**;

**default**: **throw** **new** Exception("This is not a valid data");

**endswitch**;

**else**:

$empty\_error = $key."\_empty";

$error[$empty\_error] = "**$key** cannot be empty";

**endif**;

}

**echo** "HTTP USER AGENT <br/>";

var\_dump($\_POST['server']);

**echo** "<br/>";

var\_dump($\_SERVER['HTTP\_USER\_AGENT']);

**echo** "<br/>";

**echo** "<br/>";

**echo** "TOKEN <br/>";

var\_dump($\_POST['token']);

**echo** "<br/>";

var\_dump($\_SESSION['token']);

**echo** "<br/>";

**echo** "<br/>";

**if**(isset($\_POST['submit']) && isset($\_POST['token']) && isset($\_SESSION['token'])):

**if**(

($\_SESSION['token'] == $\_POST['token'])

&& ($\_POST['server'] == $\_SERVER['HTTP\_USER\_AGENT'])

):

$clean["username"] = sanitize($\_POST['username'], "username");

$clean["password"] = sanitize($\_POST['password'], "password");

$clean["colors"] = sanitize($\_POST['colors'], "colors");

**echo** "<pre>";

var\_dump($clean["username"]);

**echo** "<br/>";

var\_dump($clean["password"]);

**echo** "<br/>";

var\_dump($clean["colors"]);

**echo** "<br/>";

**echo** "</pre>";

**endif**;

**endif**;

?>

**Code 45: process.php**

### Streams and Network Programming

The streams layer is an abstraction layer for file access. The term “stream” referes to the fact that a number of different resource—like files, but also network connections, compression protocols, and so on can be considered “streams” of data to be read and/or written either in sequence or at random. There are two types of streams. One group provides access to a certain type of stream resource; the standard PHP distribution includes several built-in examples of these:

php: Standard PHP input/output

file: Standard files access

http: Remote resources via HTTP access

ftp: Remote resouces via FTP acess

compress.zlib: Compressed data stream using the zlib compression library access

There are several other stream extensions that can be installed

#### Accessing Files

**{Handling Errors}**

##### CSV

{Networking} (File Source for the CSV File)

**fgetcsv()**

Reads the csv file

If we do not specify a delimiter and an enclosure character, both **fgetcsv()** and **fputcsv() use a comma** and **quotation marks** respectively.

<?php

**try**{

$file = fopen("networking.csv", "r");

**while**($row = fgetcsv($file)){

**echo** fgets($file)."<br/>";

}

}

**catch**(Except $e){

**echo** "Message: ".$e->getMessage();

}

**finally**{

fclose($file);

}

?>

**Code 46: opencsv.php**

**fputcsv()**

Writes in csv format

<?php

**try**{

$file = fopen("sample.csv", "w");

$showroom = **array**();

$showroom[] = ["Car", "Toytota", "Corolla"];

$showroom[] = ["Car", "Nissan", "350z"];

**foreach**($showroom **as** $car){

fputcsv($file, $car);

}

}

**catch**(Exception $e){

**echo** "Error: ".$e->getMessage();

}

**finally**{

fclose($file);

}

?>

**Code 47: writecsv.php**

##### Simple File Function

**Header(‘Content-Type: video/mpeg”);**

**Readfile(“homealone.mpeg”);**

##### Working with Directories (UNIX Based)

**chdir()**

Changes the current directory of the interpreter

**getcwd()**

Find out the current directory

**mkdir()**

Making folder

**is\_dir()**

Checks if the path is a directory

**is\_executable()**

Checks if the path is an executable

**is\_file()**

Checks if the paths exist and is a regular file

**is\_link()**

Check is the path exists and is a symlink

**is\_readable()**

Check if the path exists and is readable

**is\_writable()**

Checks if the path exists and is writable

**chmod()**

Change the moderator

#### Accessing Network Resources

##### Simple Network Access

$file = fopen("http://google.com", "r");

$page = "";

**if**($file):

**while** ($row = fread($file, 10000)):

$page .= $row;

**endwhile**;

**echo** $page;

**else**:

**throw** **new** Exception("Unable to open connection to www.google.com");

**endif**;

?>

**Code 48: simplenetwork.php**

##### Stream Contexts

Stream contexts allow us to pass options to the stream handlers that we set to access network resources, thus allowing us to tweak a handler’s behavior. We can instruct HTTP stream handler to perform a POST operation, which is very handy when we want to work with Web Services.

Stream contexts are created using **stream\_create\_context()**

**file\_get\_contents(**[**http://localhost/**](http://localhost/)**, false, stream\_create\_content())**

##### Advanced Stream Functionality

**stream\_socket\_server()**

Create socket servers

**stream\_socket\_client()**

Create socket client

<?php

$socket = stream\_socket\_server("tcp://0.0.0.0:1");

**while**($conn = stream\_socket\_accept($socket)):

fwrite($conn, "Boom Boom Shakalaka");

fclose($conn);

**endwhile**;

fclose($socket);

?>

**Code 50: socketserver.php**

<?php

$socket = stream\_socket\_client("tcp://0.0.0.0:1");

**while**(!feof($socket)):

**echo** fread($socket, 100);

**endwhile**;

fclose($socket);

?>

**Code 51: socketclient.php**

##### Stream Filters

Stream filters allow us to pass data in and out of a stream through a series of filters that can alter it dynamically, for example changing it to uppercase, passing it through a ROT-13 encoder, or compressing it suing bzip2.

**stream\_filter\_prepend()**

**stream\_filter\_append()**

### PHP Terms

**print($string)**

Exactly like echo but returns a value of 1

**Use** **pow(number, power)**

2^3 = 8

**system("python 2>&1",$output)**

Prints out what command will say in the web server

**nl2br(“\n”)**

Syntax for the new lines

**PHP\_EOL**

Create space between two numbers. EOL is End of Line

**set\_error\_handler("customError")**

**customError(error\_level,error\_message, error\_file, error\_line, error\_ context)**

This script can declare a catch-all function that is called by PHP when an error condition occurs

**ob\_start()**

Turn on output buffering

**ob\_get\_contents()**

Return the contents of the output buffer

**error\_log("Message",1,"$email","From: $email");**

Send Email About the Error

**global $var and $GLOBALS[‘var’]**

Access the global variable inside a function

**func\_get\_arg()**

Get the one argument passed int the function

**func\_get\_args()**

Get the arguments value passed in the function

**array\_shift($array)**

Remove the first element (red) from an array, and return the value of the removed element

**strlen($string)**

Measure the length of the string

**var\_dump($array)**

Prints out data with its data type for each value

**print\_r($array)**

Prints out the data from the array

**count($array)**

It counts the number of elements in an array

**is\_array($array)**

Checks whether the variable contains array or not

**array\_key\_exists($array[‘x’])**

Checks whether an array with a given key exists or not

**unset($key)**

An element can be deleted from an array by unsetting it.

**array\_flip($array)**

Inverts the value of each elements of an array with its key

**array\_reverse($array)**

Reverses the order of the elements in an array. Key association is only lost for those whose key association are numeric.

**reset($array)**

It reset the pointer to its intial position of an array

**prev($array)**

Move backward in an array

**next($array)**

Move forward in an array

**current($array)**

Current element in an array

**key($array)**

Current element’s key

**end($array)**

Start the iteration from the end of the array

**array\_walk()** and **array\_walk\_recursive()** can be used to perform an iteration of an array in which user-defined function is called.

**array\_combine($keys, $values)**

Creates an array by using one array for keys and another for its values

**strtoupper($var)**

Changes lower case letter to uppercase

**sort($array)**

It modifies the actual array. This means that we cannot call this function by passing anything other than a single variable to it.

**rsort($array), arsort($array)**

Sorts the array in reverse order

**natsort($array)**

Natural sorting.

**natcaseort($array)**

Case insensitive natural sorting.

**ksort($array)**

Sort the array using the keys

**krsort($array)**

Sort the array using the keys in reverse order

**usort($array)**

User defined sorting of an array.

**uksort($array)**

User defined sorting of an array by key.

**ursort($array)**

User defined sorting of an array by key in reverse order.

**strcmp($string, $string)**

Compares the length of the two strings

**array\_keys($array)**

Saves the keys with its value

**shuffle($array)**

Scramble the order of the array contents

**array\_push($stack, “$item”, “item”)**

Add elements at the end of the array

**array\_pop($stack)**

Remove element at the end of the array

**array\_shift($queues)**

Remove element at the start of the array

**array\_unshift($queues, “$item”)**

Add element at the start of the array

**array\_diff($array, $array)**

It is used to compute the difference between two arrays

**array\_diff\_assoc($array, $array)**

It is used to compute the difference between two arrays based on key value pairs

**array\_diff\_key($array, $array)**

It is used to compute the difference between two keys

**array\_diff\_uassoc($array, $array)**

User define differences between two arrays based on key value pairs

**array\_diff\_ukey($array, $array)**

User define difference between two keys

**array\_intersect($array, $array)**

Common section of two arrays

**array\_intersect\_key($array, $array)**

Common keys between two arrays

**array\_intersect\_assoc($array, $array)**

Common section between two arrays using key value pairs

**array\_intersect\_ukey($array, $array)**

User define common section of two array using key

**array\_intersect\_uassoc($array, $array)**

User define common section of two array using key value pair

**strstr($string, $word)**

Searches for the first occurrence of a string inside another string.

**strcmp($string, $string)**

Check whether the string matches or not, sensitive to cases.

**strcasecmp($string, $string), substr\_compare($string, $string)**

Check whether the string matches or not, insensitive to cases.

**strpos()**

It allows us to find the position of substring(***needle***), inside a string (***haystack***)

**stripos()**

It allows us to find the position of substring(***needle***), inside a string (***haystack***) (case-insensitive)

**stristr($string, $word)**

Searches for the first occurrence of a string inside another string. Then reveals everything after that. (case-insensitive)

**strspn($string, $mask, $starting\_position, $ending\_position)**

Match a string against a “whitelist” mask of allowed characteristics. It returns the length of the intial segment of the string that contains any of the characters specified in the mask.

**strcspn($string, $mask, $starting\_position, $ending\_position)**

Match a string against a “blacklist” mask of allowed characteristics.

**str\_replace($substring, $replace, $string)**

Replacing portions of a string with a different substring.

**str\_ireplace($substring, $replace, $string)**

Replacing portions of a string with a different substring. Case in-sensitive

**substr\_replace($string, $replace, $start\_location, $how\_many)**

Replacing portions of a string with a different substring.

**substr($string, $start\_location, $end\_location)**

Portions of a string

**printf($string)**

It is used to print character stream of data on stdout console.

**sprintf($string, $datatype)**

It writes a formatted string to a variable.

**fprintf($file\_path, $string, $datatype)**

It writes to a file with formatted string

**urlencode($string)**

This function is convenient when encoding a string to be used in a query part of a URL, as a convenient way to pass variables to the next page.

**$\_REQUEST**

It contains **GET, POST and cookie information**

**is\_uploaded\_file($file)**

Check whether the file is uploaded or not

**ob\_start(“ob\_hzhandler”)**

HTTP supports the transparent compression and decompression of data in transit during a transaction using the gzip algorithm. size.

**header(“Cache-Control: no-cache, must-revalidate”)**

**header(“Expires: Thu,31 May 2002 04:34:00 GMT”)**

This set of headers tells the browser not to cache the item at all by setting a cache expiration date in the past.

**get\_object\_vars($object)**

Gets the accessible non-static properties of the given object according to scope.

**spl\_autoload\_register(function($class){**

**require\_once($class.".php");**

**});**

Includes all the classes by itself

**is\_null($object)**

Checks whether the object is null or not

**is\_object($object)**

Checks whether the variable is a object or not

**get\_class()**

Returns the name of the class of an object

**str\_repeat($string, $repeat)**

It repeats a string a specified number of times.

**simplexml\_load\_string()**

Loads an XML document from a string

**simplexml\_load\_file()**

Loads an XML document from a path

**file\_get\_contents()**

The reads a file into a string.

**dom\_import\_simplexml()**

Use the xml as both **simplexml object** as well **DOM object**

**ucwords()**

It converts the first character of each word in a string to uppercase.

**ctype\_alpha($var)**

Check for alphabetic character(s)

**ctype\_alnum($var)**

Check for alphabetic character(s)

**htmlspecialchars()**

It converts some predefined characters to HTML entities.

**htmlentities()**

It converts characters to HTML entities.

**exec(), system(), passthr(),** as well as **the ` backtick operator**

Command prompt access and execution

**escapeshellcmd(), escapeshellarg()**

It escapes shell output

**fgetcsv()**

Reads the csv file

**fputcsv()**

Writes into a csv file

**stream\_create\_content()**

Tweak handlers’ behaviour

## PHP Object Oriented Analysis and Design

### Basics

Properties are also known as instance variables. A class is not an object. A class is a blueprint of an object.

### Well Designed App

Use a textual description of the problem that we are trying to solve to make sure that our desing lines up with the intended functionality of our applications.

|  |  |
| --- | --- |
| Encapsulation | It allows us to hide the inner workings of our applicaion’s parts, but yet make it clear what each part does. It keeps the parts of the code that stay the same separate from parts that change. |
| Delegation | The act of one object forwarding an operation to another object, to be performed on behalf of the first object. |
| Flexibility | It allows the software to change and grow without constant rework. It keeps our application from being fragile. |
| Design Pattern | It allows the reuse of the code and making sure we are not trying to solve a problem that someone else has already figured out |

### Gathering Requirements

|  |  |
| --- | --- |
| **External Initiator** | It kicks off the list of steps described in a use case. Without this, a use case never gets going. |
| **Use Case** | It helps us gather good requirements. Tells a story about how a system works. |
| **Start Condition** | This is always the first step in use case |
| **Requirement** | Something a system has to do to be a success |
| **Clear Value** | Without this, a use case isn’t worth anything to anyone. Uses cases without this always fail. |
| **Stop Condition** | It lets us know when a use case is finished. Without this, use cases can go on forever |
| **Main Path** | How a system work when everything is going right. This is usually what customers describe when they’re talking about the system. |
| **Requirements Change** | A complete path through a use case, from step to the last, is called a scenario. Most use cases have several different scenarious, but they always share the same user goal. |

### Analysis

It helps us ensure our system works in a real-world context. Analysis and our use cases let us show customers, managers, and other developers how our system works in a real-world context.

|  |  |
| --- | --- |
| **Terms** | **Definition** |
| Delegation | It shields our objects from implementation changes to other objects in our software. |
| Textual analysis | Looking at the noun and verbsin our use case to figure out classes and methods is called textual analysis. |
| Noun Analysis | Used in use case to figure out what classes we need in our system |
| Multiplicity | It describes how many of a specific type can be stored in the attribute of a class |
| Attribute | Equivalent to a member variable to a class |
| Class Diagram | Lists all the code-level constructs, along with their attributes and operations. |
| Operation | This is the UML term that usually represents a method in one of our classes |
| Association | It visually shows that one class has a relation to another class usually through an attribute. |
| Verb Analysis | Helps us figure out the candidates for methods on the objects in our system |

### Good Design

**Abstract classes** are placeholders for actual implementation classes. The abstract classs defines behavior, and the subclasses implement that behavior.

Whenever we find common behavior in two or more places, look to abstract that behavior into a class, and then reuse that behavior in the common classes.

**Interface**, coding to an interface, rather than to an implementation makes your software easier to extend. By coding to an interface, our code will work with all of the interface’s subclasses- even onees that haven’t been created yet.

**Cohesion**

A cohesive class does one thing really well and does not try to do or be something else. It measures the degree of connectivity among the elements of a single module, class or object. The higher the cohesion of our software is, the more well-defined and realted the responsibilities of each individual class in our application. Each class has a very specific set of closely related actions it performs.

**Domain Analysis**

It let us check our designes and still speak the customer’s language. The process of identifying, collecting, organizing, and representing the relevant information of a domain, baase dupon the study of existing systems and their development histories, knowledge captures from domain experts, underlying theory, and emerging technology within a domain.

### Architecture

It is our design structure, and highlights the most important parts of our app, and the relationships between those parts. It is the organizational structure of a system, including its decompoisiton into parts, their connectivity, interaction mechanisms, and the guiding principles and decisions that we use in the design of a system.

### Design Principle

It is a basic tool or technique that can be applied to designing or writing code to make that code more maintainable, flexible, or extensible.

* **Open-Closed Principle (OCP):** Classe should be open for extension, and closed for modification.
* **Don’t Repeat Yourself (DRY):** Avoid duplicate code abstracting out things that are common and placing those thins in a single location. It is aobut having each piece of information and behavior in our systems in a single, sensible place.
* **Single Responsibilit Principle (SRP):** Every object in our system should have a single responsibility, and all the object’s service should be focused on carrying out that single responsibility.
* **The Liskov Substitution Principle (LSP):** Subtypes must be substitutable for their base types.

**Composition**

It allows us to use behavior from a family of other classes, and to change that behavior at runtime. In composition, the object composed of other behaviours owns those behaviours. When the object is destroyed, so are all of its behaviors. The behaviors in a composition do not exist outside of the composition itself. Represent by a line with a closed diamond

**≪Interface≫**

Weapons

attack()

**Unit**

type

***properties***

setType

getType

setProperty

getProperty

**Club**

attack()

**Gun**

attack()

**Sword**

attack()

**Fig 1:** **Composition UML Diagram**

**Aggregation**

It is when a one class is used as part of another class, but still exists outside of that other class. Represent by an open diamond

InstrumentSpec

properties

getProperty()

getProperties()

matches

**Instrument**

serialNumber

price

getSerialNumber

getPrice

setPrice

***getSpec***

**Fig 2:** **Aggregation UML Diagram**

|  |  |  |
| --- | --- | --- |
| **Delegation** | **Composition** | **Aggregation** |
| Delegate behaviours to another class when we don’t want to change the behavior to implement that behavior on its own | We can reuse behavior from one or more classes, and in particular from a family of classes, with composition. Our object completely owns the composed objects, and they do not exist outside of their usage in our object | When we want the benefits of composition but we are using the behavior from an object that does exist outside of our objects |

### Testing

Test driven development focuses on getting the behavior of our classes right.

When we are programming to **contract**, we are working with client code to agree on how we will handle problem situtations. Programing by contract sets up an agreement about how our software behaves that we and users of our software agree to abide by.

When we are programming **defensively**, you are making sure the client gets a “safe” response, no matter what the clients wants to have happen. Defensive programming doesn’t trust other software, and doe sextensive error and data checking to ensure the other software doesn’t give us bad or unsafe information.

**Development Approaches**

*Use Case Driven Development:* It takes a single use case in our system, and focuses on completing the code to implement that entire use case, including all of its scenarios, before moving on to anything else in the applicaton.

*Feature-driven development:* It focuses on a single feature, and codes all the behavior of that feature, before moving on to anything else in the application.

*Test driven development:* It writes test scenarious for a peiece of functionality before writing the code for that functionality. Then we write software to pass all the texts.

Good software development usually incorporates all of these development models at different stages of the development cycle

### Software Development Cycle

|  |  |
| --- | --- |
| Feature List | Figure out what our app is supposed to do at a higher level |
| Use Case Diagrams | Nail down the big processes that our app performs, and any external forces that are involved |
| Break Up the Problem | Break our application up into modules of functionality, and then decide on an order in which to tackle each of our modules |
| Requirements | Figure out the individual requirements for each module, and make sure those fit in with the big picture. |
| Domain Analysis | Figure out how our use cases map to objects in our app, and make sure our customer is on the same as we are |
| Preliminary Design | Fill in details about our objects, define relationships between the objects and apply principles and patterns |
| Implementation | Write code, test it, and make sure it works. Do this for each behavior, each feature, each use case, each problem, until we are done |
| Delivery | Release the software. |

### Anti Pattern

|  |  |
| --- | --- |
| **Design Patterns** | **Anti Patterns** |
| Design patterns are proven solutions to particular types of problems, and help us structure our own applications in way that are easier to understand, more maintainable, and more flexible | Anti-patterns are the reverse of design patterns: they are common BAD solutions to problems, these dangerous pitfalls should be recognized and avoided. |

### Extras

##### CRC

It stands for Class, Responsibility, Collaborator. (Similar flash cards)

##### Metrics

More details in Agile Software Development

**Refactoring**

It is the process fo modifying the structure of our code without modifying its behavior. Refactoring changes the internal structure of our code without affecting our code’s behavior.

**Unified Modeling Language**

UML can be represented by State diagrams, Sequence diagram and Class Diagrams. It is a language used to communicate just the details about our code and application’s structure that other developers and customers need, without getting details that aren’t necessary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Inheritance | When one class inherits behavior from another class, and can then change the behavior if needed |
| Subclass | The class that extends from the parent class; Child Class |
| Supertype/Superclass | The subclass that inherited behaviours from |
| Method Overriding | The subclass can change the behavior of its superclass, as well as call the superclass’s methods. This is called overriding the superclass’s behavior |
| Polymorphism | It is closely related to inheritance. When one class inherits from another, then polymorphism allows a subclass to stand in for the superclass |

|  |  |
| --- | --- |
| **Method Overriding** | **Method Overloading** |
| * Arguments must be the same, return type must also be compatible. * The methods cannot be less accessible | * The return types can be different * They have different arguments list * You can’t just change the return type. Both argument and return types need to be changed * You can vary the access level in any direction |

**Encapsulation,** it helps us protect our classes from unnecessary changes and encapsulate what varies.

Apply basic Obejct Oriented principles to add flexibility. By encapsulating what varies, our application is more flexible, and easier to change. When we have a set of properties that vary across our objects, use array to store those properties dynamically. This will remove lots of methods from our classes, and avoid having to change our code when new properties are added to our app.

The process of enclosing programming elements inside larger, more abstract entities. Also know as information hiding or separation of concerns.

### Object Oriented Principles

* Encapsulate what varies
* Code to an interface rather than to an implementation
* Each class in our application should have only one reason to change
* Classes are about behavior and functionality
* Classes should be open for extension, but closed for modification (OCP)
* Avoid Duplicate code by abstracting out things that are common and placing them in a single location (DRY)
* Every object in our system should have a single responsibility, and all the object’s services should be focused on carrying out that single responsibility (SRP)
* Subclasses should be suitable for their base classes (LSP)

## PHP Design Pattern

### UML Diagram

***InstrumentSpec***

model:String

getBuilder(): Builder

getModel(): String

getType(): Type

getBackWood(): Wood

getTopWood(): Wood

matches(InstrumentSpec): boolean

***Instrument***

serialNumber: String

price: double

getSerialNumber(): String

getPrice(): double

setPrice(float)

getSpec(): InstrumentSpec

**Sitar**

**Guitar**

**Fig 1: Basic UML Diagram**

#### UML Cheat Sheet

|  |  |  |
| --- | --- | --- |
| **Pattern** | **UML Term** | **UML Diagram** |
| Interface | Interface | **≪** interface**≫** |
| Abstract Class | Abstract Class | *Italicized Class Name* |
| Relationship | Association |  |
| Inheritance | Generalization |  |
| Aggregation | Aggregation |  |

**UML: Unified Modeling Language**

### Strategy Pattern

**Design Principle:** Identify the aspects of our application that vary and separate them from what stays the same.

**Design Principle:** Program to an interface, not to an implementation

**Design Principle:** Favor composition over inheritance

Take what varies and “encapsulate” it so it won’t affect the rest of our code. Separate the behaviours (methods) and make them into a separate class.

When we put two or more behavioural classes together, this is known as composition. It lets use change the behavior at runtime.

**IS-A** refers to **inheritance**, **HAS-A** refers to **composition and aggregation**

**Definition:** It defines a family of algorithms, excapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

**Client**

**Encapsulated Superpower Behaviour**

***Superhero***

superpower: **superpower**

gadgets**: gadgets**

name(): NULL

power(): NULL

gadgetuse(): NULL

display(): NULL

**≪**interface**≫**

**superpower**

power()

**Laser**

power(){}{}

**Web**

power(){}

**NoSuperPower**

power(){}

**Encapsulated Superpower Behaviour**

**≪**interface**≫**

**gadgets**

gadgetuse()

**Spiderman**

name(){}

**Superman**

Name(){}

**Batman**

name(){}

**web**

gadgetuse(){}

**nosuperpower**

gadgetuse(){}

**laser**

gadgetuse(){}

**Fig 2: Strategy Pattern UML Diagram**

<?php

*/\*\**

*\* Interface Superpower*

*\*/*

**interface** superpower{

**public** **function** power();

}

*/\*\**

*\* Web one of the superpower*

*\*/*

**class** **web** **implements** superpower{

**public** **function** power(){

**return** "I can throw webs!";

}

}

*/\*\**

*\* Laser one of the superpower*

*\*/*

**class** **laser** **implements** superpower{

**public** **function** power(){

**return** "I can shoot ray!";

}

}

*/\*\**

*\* No Superpower*

*\*/*

**class** **nosuperpower** **implements** superpower{

**public** **function** power(){

**return** "I have no superpower!";

}

}

*/\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

*/\*\**

*\* Interface Gadgets*

*\*/*

**interface** gadgets{

**public** **function** gadgetuse();

}

*/\*\**

*\* Batarang one of the gadgets*

*\*/*

**class** **batarang** **implements** gadgets{

**public** **function** gadgetuse(){

**return** "I can throw my batarang!";

}

}

*/\*\**

*\* No Gadgets*

*\*/*

**class** **nogadgets** **implements** gadgets{

**public** **function** gadgetuse(){

**return** "I do not use any gadget!";

}

}

*/\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

*/\*\**

*\* Abstract Superhero*

*\*/*

**abstract** **class** **superhero**{

**protected** $superpowers;

**protected** $gadgets;

**abstract** **public** **function** name();

**public** **function** power(){

**return** $this->superpowers->power();

}

**public** **function** gadgetuse(){

**return** $this->gadgets->gadgetuse();

}

**public** **function** display(){

**echo** nl2br("<strong>".$this->name()."</strong>**\n**") ;

**echo** nl2br("Superpower: ".$this->power()."**\n**");

**echo** nl2br("Gadget: ".$this->gadgetuse()."**\n**");

**echo** nl2br("--------------------------------------------------**\n**");

}

}

*/\*\**

*\* Spiderman one of the superheroes*

*\*/*

**class** **Spiderman** **extends** superhero{

**public** **function** \_\_construct(){

$this->superpowers = **new** web();

$this->gadgets = **new** nogadgets();

}

**public** **function** name(){

**return** get\_class($this);

}

}

*/\*\**

*\* Superman one of the superheroes*

*\*/*

**class** **Superman** **extends** superhero{

**public** **function** \_\_construct(){

$this->superpowers = **new** laser();

$this->gadgets = **new** nogadgets();

}

**public** **function** name(){

**return** get\_class($this);

}

}

*/\*\**

*\* Batman one of the superheroes*

*\*/*

**class** **Batman** **extends** superhero{

**public** **function** \_\_construct(){

$this->superpowers = **new** nosuperpower();

$this->gadgets = **new** batarang();

}

**public** **function** name(){

**return** get\_class($this);

}

}

*/\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

*/\*\**

*\* Testing*

*\*/*

$spiderman = **new** Spiderman();

$spiderman->display();

$superman = **new** Superman();

$superman->display();

$batman = **new** Batman();

$batman->display();

?>

**Code 1: strategy.php**

### Observer Pattern

Analaogy to newspaper subscription, job searching

**Publishers + Subscribers = Observers Pattern**

Over here, the **publishers** are the **subject** and subscribers are the **observers**

**Definition:** It defines a one-to-many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically. It defines a one-to-many relationship between a set of objects. When the state of one object changes, all of its dependents are notified

**Loosely coupled**

When two objects are loosely coupled, they can interact, but have very little knowledge of each other. The observer pattern provides an object desing where subjects and observers are loosely coupled. Loosely coupled designs allow us to build flexible Object Oriented systesm that can handle change because they minimize the interdependency between objects. Strive for loosely coupled designs between objects that interact.

**array\_search($value, $array)**

It searches an array for a value and returns the key.

**array\_values($array)**

It returns an array containing all the values of an array.

**≪Interface≫**

**Observer**

update($s, $c, $r)

**Desktop**

update($s, $c, $r)

display()

**≪Interface≫**

**Subject**

registerObserver(observer)

removeObserver(observer)

notifyObserver()

**Tablet**

update($s, $c, $r)

display()

**≪Interface≫**

**Device**

display()

**SmartWatch**

Observers[]: Observer

registerObserver(observer)

removeObserver(observer)

notifyObserver()

dataChange($s, $c. $r)

**SmartPhone**

update($s, $c, $r)

display()

**Fig 3: Observer Pattern UML Diagram**

<?php

*/\*\**

*\* Subject (Health Fitness Measurement)*

*\*/*

**interface** subject{

**public** **function** registerObserver(Observer $observer);

**public** **function** removeObserver(Observer $observer);

**public** **function** notifyObserver();

}

**class** **smartwatch** **implements** subject{

**public** $observers;

**protected** $steps;

**protected** $calories;

**protected** $rest;

**public** **function** \_\_construct(){

$this->observers = **array**();

}

**public** **function** registerObserver(Observer $observer){

$this->observers[] = $observer;

}

**public** **function** removeObserver(Observer $observer){

$key = array\_search($observer, $this->observers);

unset($this->observers[$key]);

$this->observers = array\_values($this->observers);

}

**public** **function** notifyObserver(){

**for**($i = 0; $i < count($this->observers); $i++){

$this->observers[$i]->update($this->steps, $this->calories, $this->rest);

}

}

**public** **function** dataChange($steps, $calories, $rest){

$this->steps = $steps;

$this->calories = $calories;

$this->rest = $rest;

$this->notifyObserver();

}

}

*//=========================================================================*

*//=========================================================================*

*/\*\**

*\* Observer (Devices)*

*\*/*

**interface** observer{

**public** **function** update($steps, $calories, $rest);

}

**interface** device{

**public** **function** display();

}

**class** **tablet** **implements** observer, device{

**protected** $steps;

**protected** $calories;

**protected** $rest;

**public** **function** update($steps, $calories, $rest){

$this->steps = $steps;

$this->calories = $calories;

$this->rest = $rest;

$this->display();

}

**public** **function** display(){

**echo** "Pedometer: ".$this->steps."<br/>";

**echo** "Eating: ".$this->calories."<br/>";

**echo** "Sleep Hours: ".$this->rest."<br><br/>";

}

}

**class** **smartphone** **implements** observer, device{

**protected** $steps;

**protected** $calories;

**protected** $rest;

**public** **function** update($steps, $calories, $rest){

$this->steps = $steps;

$this->calories = $calories;

$this->rest = $rest;

$this->display();

}

**public** **function** display(){

**echo** "Pedometer: ".$this->steps."<br/>";

**echo** "Eating: ".$this->calories."<br/>";

**echo** "Sleep Hours: ".$this->rest."<br/><br/>";

}

}

**class** **desktop** **implements** observer, device{

**protected** $steps;

**protected** $calories;

**protected** $rest;

**public** **function** update($steps, $calories, $rest){

$this->steps = $steps;

$this->calories = $calories;

$this->rest = $rest;

$this->display();

}

**public** **function** display(){

**echo** "Pedometer: ".$this->steps."<br/>";

**echo** "Eating: ".$this->calories."<br/>";

**echo** "Sleep Hours: ".$this->rest."<br/><br/>";

}

}

*//=========================================================================*

*//=========================================================================*

*/\*\**

*\* Testing*

*\*/*

*/\*Observers \*/*

$tablet = **new** tablet();

$desktop = **new** desktop();

$smartphone = **new** smartphone();

*/\*Subject\*/*

$smartwatch = **new** smartwatch();

*/\*Register the Observers\*/*

$smartwatch->registerObserver($tablet);

$smartwatch->registerObserver($desktop);

$smartwatch->registerObserver($smartphone);

$smartwatch->dataChange(34.56, 234.9, 234.34);

*/\*Remove the Observer \*/*

$smartwatch->removeObserver($desktop);

**echo** "<pre>";

var\_dump($smartwatch->observers);

**echo** "</pre>";

$smartwatch->dataChange(23.21, 123.9, 98.1);

?>

**Code 2: observer.php**

### Decorator Pattern

**Open Closed Principle:** Classes should be open for extension, but closed for modification.

Our goal is to allow classes to be easily extended to incorporate new behavior without modifying existing code.

**Definition:** It attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.

**Design Principle:** Classes should be open for extension, but closed for modification

***Item***

Description

Extras

getDescription()

cost() [ABSTRACT]

**French Fries**

**Cheese**

**Chicken Burger**

**Drink**

**Fig 4: Decorator Pattern UML Diagram**

<?php

**abstract** **class** **Item**{

**protected** $description = "";

**protected** $extras = **array**();

**public** **function** getDescription(){

**return** $this->description;

}

**abstract** **function** cost();

}

**class** **ChickenBurger** **extends** Item{

**public** **function** \_\_construct($extras = **NULL**){

$this->extras = $extras;

$this->description = "Chicken Burger";

}

**public** **function** getDescription(){

$items = **array**();

**if**($this->extras == **NULL**):

**return** $this->description;

**else**:

$this->description.= " with ";

**if**(is\_array($this->extras)):

**foreach**($this->extras **as** $extra):

$items[] = $extra->getDescription();

**endforeach**;

$totalitems = implode(", ", $items);

**return** $this->description." **{**$totalitems**}**";

**else**:

**return** $this->description." **{**$this->extras->getDescription()**}**";

**endif**;

**endif**;

}

**public** **function** cost(){

$cost = 0;

**if**($this->extras == **NULL**):

**return** 250;

**else**:

**if**(is\_array($this->extras)):

**foreach**($this->extras **as** $extra):

$cost = $extra->cost() + $cost;

**endforeach**;

**return** 250 + $cost;

**else**:

**return** 250 + $this->extras->cost();

**endif**;

**endif**;

}

}

**class** **frenchfries** **extends** Item{

**public** **function** \_\_construct(){

$this->description = "French Fries";

}

**public** **function** cost(){

**return** 100;

}

}

**class** **cheese** **extends** Item{

**public** **function** \_\_construct(){

$this->description = "Cheese";

}

**public** **function** cost(){

**return** 50;

}

}

**class** **drink** **extends** Item{

**public** **function** \_\_construct(){

$this->description = "Drink";

}

**public** **function** cost(){

**return** 25;

}

}

$frenchfries = **new** frenchfries();

$cheese = **new** cheese();

$drink = **new** drink();

$extras = [$frenchfries, $cheese, $drink];

$chickenburger = **new** chickenburger([$frenchfries, $drink]);

**echo** $chickenburger->getDescription()."<br/>";

**echo** "Cost: ".$chickenburger->cost()."<br/>";

?>

**Code 3: decorator.php**

### Factory Method Pattern

**Definition:** It defines an interface for creating an object, but lets subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.

**print\_r($array, true)**

Instead of printing out, it will save to a variable.

**Design Principle**: Depend upon abstractions. Do not depend upon concrete classes.

***≪Interface≫***

**arraysave**

create($filename, $array)

***filewritetext***

create($array)

***≪Interface≫***

**type**

create($array)

***filewritejson***

create($array)

***fileputjson***

create($array)

***fileputcontents***

***filewrite***

***fileputtext***

create($array)

**Fig 5: Factory Method Pattern UML Diagram**

<?php

*/\*\**

*\* Creator*

*\*/*

**interface** arraysave{

**public** **function** create($filename, $array);

}

**class** **filewrite** **implements** arraysave{

**public** **function** create($filename, $array){

$separate = explode(".", $filename);

**switch**($separate[1]):

**case** "json":

$json = **new** filewritejson();

**return** $json->create($filename, $array);

**break**;

**case** "txt":

$text = **new** filewritetext();

**return** $text->create($filename, $array);

**break**;

**default**: **throw** **new** Exception("Now a Valid Filename");

**endswitch**;

}

}

**class** **fileputcontents** **implements** arraysave{

**public** **function** create($filename, $array){

$separate = explode(".", $filename);

**switch**($separate[1]):

**case** "json":

$json = **new** fileputjson();

**return** $json->create($filename, $array);

**break**;

**case** "txt":

$text = **new** fileputtext();

**return** $text->create($filename, $array);

**break**;

**default**: **throw** **new** Exception("Now a Valid Filename");

**endswitch**;

}

}

*/\*\**

*\* Product*

*\*/*

**Interface** type{

**public** **function** convert($array);

}

**class** **filewritejson** **implements** type{

**public** **function** create($filename, $array){

**try**{

$file = fopen($filename, "w");

fwrite($file, $this->convert($array));

}

**catch**(Exception $e){

**echo** "Error: ".$e->getMessage();

}

**finally**{

fclose($file);

}

}

**public** **function** convert($array){

**return** json\_encode($array);

}

}

**class** **filewritetext** **implements** type{

**public** **function** create($filename, $array){

**try**{

$file = fopen($filename, "w");

fwrite($file, $this->convert($array));

}

**catch**(Exception $e){

**echo** "Error: ".$e->getMessage();

}

**finally**{

fclose($file);

}

}

**public** **function** convert($array){

**return** print\_r($array, **true**);

}

}

**class** **fileputjson** **implements** type{

**public** **function** create($filename, $array){

**return** file\_put\_contents($filename, $this->convert($array));

}

**public** **function** convert($array){

**return** json\_encode($array);

}

}

**class** **fileputtext** **implements** type{

**public** **function** create($filename, $array){

**return** file\_put\_contents($filename, $this->convert($array));

}

**public** **function** convert($array){

**return** print\_r($array, **true**);

}

}

*/\*\**

*\* Testing*

*\*/*

$array = ["India"=>"Hindi", "America" => "English", "France" => "French"];

*/\*Creating File Using File Write\*/*

$filewrite = **new** filewrite();

$filewrite->create("language.json", $array);

$filewrite->create("language.txt", $array);

*/\*Creating File Using File Put Contents\*/*

$fileput = **new** fileputcontents();

$fileput->create("lang.json", $array);

$fileput->create("lang.txt", $array);

?>

**Code 4: factory.php**

**Dependency Inversion Principle:** Depend upon abstractions. Do not depend upon concrete classes. High-level component should not depend on our low-level component components, rather, they should both depend on abstractions.

A few guidelines to help us follow the principle

* No variable should hold a reference to a concrete class
* No class should derive from a concrete class
* No method should override an implemented method of any of its base classes.

### Abstract Factory Pattern

**Definition:** It provides an interface for creating families of related or dependent objects without specifying their concrete classes.

***≪Interface≫***

**Factory**

getShape()

getColor()

**ShapeFactory**

**ColorFactory**

***≪Interface≫***

**Shape**

draw()

***≪Interface≫***

**Color**

fill()

**Ellipse**

**Rectangle**

**Blue**

**Red**

**Fig 6: Abstract Factory Pattern UML Diagram**

<?php

*/\*\**

*\* Shape Interface*

*\*/*

**interface** Shape{

**public** **function** draw();

}

**class** **Rectangle** **implements** Shape{

**public** **function** draw(){

**return** "Rectangle";

}

}

**class** **Ellipse** **implements** Shape{

**public** **function** draw(){

**return** "Ellipse";

}

}

*/\*\**

*\* Color Interface*

*\*/*

**interface** Color{

**public** **function** fill();

}

**class** **Red** **implements** Color{

**public** **function** fill(){

**return** "Red";

}

}

**class** **Blue** **implements** Color{

**public** **function** fill(){

**return** "Blue";

}

}

*/\*\**

*\* Abstract Factory Interface*

*\*/*

**Interface** AbstractFactory{

**public** **function** getShape($shape);

**public** **function** getColor($color);

}

**class** **ShapeFactory** **implements** AbstractFactory{

**public** **function** getShape($shape){

$shape = strtolower($shape);

**switch**($shape):

**case** "rectangle":

$rectangle = **new** Rectangle();

**return** $rectangle->draw();

**break**;

**case** "ellipse":

$ellipse = **new** Ellipse();

**return** $ellipse->draw();

**break**;

**default**: **throw** **new** Exception("Shape not recognized");

**endswitch**;

}

**public** **function** getColor($color){

**return** **null**;

}

}

**class** **ColorFactory** **implements** AbstractFactory{

**public** **function** getShape($shape){

**return** **null**;

}

**public** **function** getColor($color){

$color = strtolower($color);

**switch**($color):

**case** "red":

$red = **new** Red();

**return** $red->fill();

**break**;

**case** "blue":

$blue = **new** Blue();

**return** $blue->fill();

**break**;

**default**: **throw** **new** Exception("Color not recognized");

**endswitch**;

}

}

*/\*\**

*\* Factory Producer*

*\*/*

**class** **FactoryProducer**{

**public** **function** \_\_construct($shape, $color){

$colorFactory = **new** ColorFactory();

$shapeFactory = **new** ShapeFactory();

**echo** $colorFactory->getColor($color)." "

.$shapeFactory->getShape($shape)."<br/>";

}

}

*/\*\**

*\* Testing*

*\*/*

**echo** nl2br("Color + Shape = Filled Colors**\n\n**");

$coloredshape = **new** FactoryProducer("rectangle","red");

$coloredshape = **new** FactoryProducer("Rectangle","blue");

$coloredshape = **new** FactoryProducer("ellipse","blue");

$coloredshape = **new** FactoryProducer("Ellipse","red");

?>

**Code 5: abstractfactory.php**

### Singleton Pattern

**Definition:** It ensures a class has only one instance, and provides a global point of access to it.

<?php

**class** **database**{

**private** **static** $database;

**private** **static** $attempt = 0;

**private** **function** \_\_construct(){}

**public** **static** **function** connect(){

**if**(is\_null(self::$database)):

self::$database = **new** database();

**echo** "Database Connected<br/>";

**else**:

self::$attempt += 1;

**echo** "Database Already Connected, Attempt Failure: ".self::$attempt."<br/>";

**endif**;

**return** self::$database;

}

}

*/\*\**

*\* Testing*

*\*/*

$database = database::connect();

$database2 = database::connect();

$database3 = database::connect();

$database4 = database::connect();

$database5 = database::connect();

?>

**Code 6: singleton.php**

### Command Pattern

The client creates a command object. The client does a **setCommand()** to store the command object in the invoker. The client asks the invoker to execute to execute the command.

**Definition:** It encapsulates a request as an object, thereby letting us parametrize other objects with different requests, queue or log requests, and support undoable operations.

An **invoker** makes a request of a Command object by calling its **execute()** method, which invokes those actions on the receiver. Invokers can be parameterized with Commands, even dynamically at runtime.

**Macro Commands** are a simple extension of Command that allow multiple commands to be invoked. Likewise, Macro commands can easily support **undo().** Commands may also be used to implement loggingand transactional systems.

**Remote**

setCommand()

buttonPushed()

***≪Interface≫***

**Command**

execute()

**GarageClose**

**GarageOpen**

**LightOff**

**LightOn**

**Garage**

open()

close()

**Light**

on()

off()

**Fig 7: Command Pattern UML Diagram**

<?php

*/\*\**

*\* Object*

*\*/*

**class** **Light**{

**public** **function** on(){

**return** "Light on<br/>";

}

**public** **function** off(){

**return** "Light off <br/>";

}

}

**class** **GarageDoor**{

**public** **function** open(){

**return** "Door Open<br/>";

}

**public** **function** close(){

**return** "Door Close<br/>";

}

}

*/\*\**

*\* Command Interface*

*\*/*

**interface** Command{

**public** **function** execute();

}

**class** **LightOnCommand** **implements** Command{

**protected** $light;

**public** **function** \_\_construct($light){

$this->light = $light;

}

**public** **function** execute(){

**return** $this->light->on();

}

}

**class** **LightOffCommand** **implements** Command{

**protected** $light;

**public** **function** \_\_construct($light){

$this->light = $light;

}

**public** **function** execute(){

**return** $this->light->off();

}

}

**class** **DoorOpenCommand** **implements** Command{

**protected** $garage;

**public** **function** \_\_construct($garage){

$this->garage = $garage;

}

**public** **function** execute(){

**return** $this->garage->open();

}

}

**class** **DoorCloseCommand** **implements** Command{

**protected** $garage;

**public** **function** \_\_construct($garage){

$this->garage = $garage;

}

**public** **function** execute(){

**return** $this->garage->close();

}

}

*/\*\**

*\* Remote Control*

*\*/*

**class** **RemoteControl**{

**protected** $commands;

**public** **function** setCommand(){

$num = func\_num\_args();

$data = func\_get\_args();

**for**($i = 0; $i < $num; $i++):

$this->commands[] = $data[$i];

**endfor**;

}

**public** **function** buttonPushed(){

**foreach**($this->commands **as** $command):

**echo** $command->execute();

**endforeach**;

}

}

*/\*\**

*\* Turning on the light*

*\*/*

$light = **new** Light();

$lighton = **new** LightOnCommand($light);

*/\*\**

*\* Opening Garage Door*

*\*/*

$garage = **new** GarageDoor();

$garageopen = **new** DoorOpenCommand($garage);

*/\*\**

*\* Macro Command*

*\*/*

$remote = **new** RemoteControl();

$remote->setCommand($lighton, $garageopen);

$remote->buttonPushed();

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*0

*/\*\**

*\* Closing Garage Door*

*\*/*

$garageoff = **new** DoorCloseCommand($garage);

*/\*\**

*\* Turning off the light*

*\*/*

$lightoff = **new** LightOffCommand($light);

*/\*\**

*\* Macro Command*

*\*/*

$remote = **new** RemoteControl();

$remote->setCommand($lightoff, $garageoff);

$remote->buttonPushed();

?>

**Code 7: command.php**

### Adapter Pattern

The client makes a request to the adapter by calling a method on it using the target interface. The adapter translates that request into one or more calls on the adaptee using the adaptee interface. The client receives the results of the call and never knows there is an adapter doing the translation.

**Definition:** It converts the interface of a class into another interface the clients expect. Adapter lets classes work together that couldn’t otherwise because of incompatible interfaces. The **object adapters** and **class adapters** are two types of adapter pattern.

**spotify**

spotifymusic()

**ipod**

applemusic()

**spotifyAdapter**

spotify

applemusic()

**Fig 8: Class Adapters UML Diagram**

<?php

**class** **ipod**{

**public** **function** applemusic(){

**echo** "Ipod Music";

}

}

**class** **spotify**{

**public** **function** spotifymusic(){

**echo** "Spotify Music";

}

}

**class** **spotifyAdapter**{

**protected** $spotify;

**public** **function** \_\_construct(spotify $spotify){

$this->spotify = $spotify;

}

**public** **function** applemusic(){

$this->spotify->spotifymusic();

}

}

$adapter = **new** spotifyAdapter(**new** spotify());

$adapter->applemusic();

?>

**Code 8: classadapter.php**

**≪Interface≫**

**musicplayer**

playaudio()

**≪Interface≫**

**videoplayer**

playvideo()

**powerdvd**

**powerdvdAdapter**

videoplayer

**mp4player**

**winamp**

**Fig 9: Object Adapters UML Diagram**

<?php

**interface** musicplayer{

**public** **function** playaudio();

}

**class** **winamp** **implements** musicplayer{

**public** **function** playaudio(){

**echo** "Winamp music";

}

}

**class** **mp4player** **implements** musicplayer{

**public** **function** playaudio(){

**echo** "Mp4player music";

}

}

**interface** videoplayer{

**public** **function** playvideo();

}

**class** **powerdvd** **implements** videoplayer{

**public** **function** playvideo(){

**echo** "Powerdvd video";

}

}

**class** **powerdvdAdapter** **implements** musicplayer{

**protected** $videoplayer;

**public** **function** \_\_construct(videoplayer $videoplayer){

$this->videoplayer = $videoplayer;

}

**public** **function** playaudio(){

**return** $this->videoplayer->playvideo();

}

}

**Code 9: objectadapter.php**

### Facade Pattern

A facade not only simplifies an interface, it decouples a client from a subsystem of components. Facades and adapters mey wray multiple classes, but facade’s intent is to simplify, while an adapter’s is to convert the interface to something different

**Definition:** It provides a unified interface to a set of interfaces in a subsustem. Façade defines a higher-level interface that makes thhe subsystem easier to use.

**Design Principle**: Principle of Least Knowledge – talk only to our immediate friends.

**shapeMaker**

rectangle

ellipse

drawRectangle()

drawEllipse()

**≪Interface≫**

**Shape**

draw()

**Rectangle**

**Ellipse**

**Fig 9: Façade Pattern UML Diagram**

<?php

**interface** shape{

**public** **function** draw();

}

**class** **rectangle** **implements** shape{

**public** **function** draw(){

**echo** "Rectangle"."<br/>";

}

}

**class** **ellipse** **implements** shape{

**public** **function** draw(){

**echo** "Ellipse"."<br/>";

}

}

**class** **shapeMaker**{

**protected** $rectangle;

**protected** $ellipse;

**public** **function** \_\_construct(){

$this->rectangle = **new** rectangle();

$this->ellipse = **new** ellipse();

}

**public** **function** drawRectangle(){

**return** $this->rectangle->draw();

}

**public** **function** drawEllipse(){

**return** $this->ellipse->draw();

}

}

$shapeMaker = **new** shapeMaker();

$shapeMaker->drawRectangle();

$shapeMaker->drawEllipse();

?>

**Code 10: facade.php**

### Template Method Pattern

The template method defines the steps of an algorithm and allows sublclasses to provide the implementation for one or more steps.

**Definition:** It defines the skeleton of an algorithm in a method, deferring some steps to subclasses. Template method lets subclasses redefine certain steps of an algorithm without changing the algorithm’s structure.

Concrete methods that do nothing by default are called hooks. Subclasses are free to override these. A **hook** is a method that is declared in the abstract class, but only given an empty or default implementation.

**Design Principle:** Don’t call us, we will call you.

***HTMLTemplate***

prepare() **final**

header()

body()

titleContent() [ABS]

external() [ABS]

bodyContent() [ABS]

footer() [ABS]

**HomePage**

titleContent()

external()

bodyContent()

footer()

**Fig 9: Template Pattern UML Diagram**

<?php

**abstract** **class** **HTMLTemplate**{

**public** **final** **function** prepare(){

$this->header();

$this->titleContent();

$this->external();

$this->body();

$this->bodyContent();

$this->footer();

}

**public** **function** header(){

**echo** "<!DOCTYPE html>";

**echo** "<html>";

**echo** " <head>";

**echo** " <title>".$this->titleContent()."</title>";

**echo** " <meta charset = 'UTF-8'>";

$this->external();

**echo** " </head>";

}

**public** **function** body(){

**echo** " <body>";

**echo** $this->bodyContent();

**echo** " </body>";

**echo** "<footer>".$this->footer()."</footer>";

**echo** "</html>";

}

**abstract** **function** titleContent();

**abstract** **function** external();

**abstract** **function** bodyContent();

**abstract** **function** footer();

}

**class** **HomePage** **extends** HTMLTemplate{

**public** **function** titleContent(){

**return** "Home Page";

}

**public** **function** external(){

}

**public** **function** bodyContent(){

**return** "<p> Home Page Bro </p>";

}

**public** **function** footer(){

**return** "&copy;".date("Y")." All Rights Reserved. Photon Enterprise";

}

}

$homepage = **new** HomePage();

$homepage->prepare();

?>

**Code 11: template.php**

### Iterator Pattern

**Definition:** It provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

It also places the task of traversal on the iterator object, not on the aggregate, which simplifies the aggregate interface and implementation, and places the responsibility where it should be.

**Single Responsibility**

Every responsibility of a class is an area of potential change. More than one responsibility menas more than one are of change. This principle guides us to keep each class to a single responsibility.

**Design Principle:** A class should have only one reason to change

**Cohesion:** It is a term we would hear used as a measure of how closely a class or a module supports a single purpose or responsibility. We say that a module or class has high cohesion when it is designed around a set of related functions, and we sat it has low cohesion when it is designed around a set of unrelated functions. Classes that tend to have cohesion and are more maintainable than classes take on multiple responsibilities and have low cohesion.

**Collections Type Heirarchy (Important Ones)**

🡪Iterable

* Array
* Transversable
  + Iterator
    - Generator
  + IteratorAggregate

**SPL Iterators Class Tree (Important Ones)**

ArrayIterator

* RecursiveArrayIterator

IteratorIterator

* FilterIterator

RecursiveIteratorIterator

**≪Interface≫**

**Iterator**

current()

next()

rewind()

key()

valid()

**Library**

books

bookCount()

getBook()

addBook()

removeBook()

**Book**

title

author

getTitle()

getAuthor()

getTitleAuthor()

**≪Interface≫**

**BookListIterator**

library

location

prev()

**Fig 10: Iterator Pattern UML Diagram**

<?php

**class** **Book**{

**protected** $title;

**protected** $author;

**public** **function** \_\_construct($title, $author){

$this->title = $title;

$this->author = $author;

}

**public** **function** getTitle(){

**return** $this->title;

}

**public** **function** getAuthor(){

**return** $this->author;

}

**public** **function** getTitleAuthor(){

**return** "Title: ".$this->title.", Author: ".$this->author;

}

}

**class** **Library**{

**protected** $books;

**public** **function** bookCount(){

**return** count($this->books);

}

**public** **function** getBook($number){

**if**(array\_key\_exists($number, $this->books)):

**return** $this->books[$number];

**else**:

**echo** "Book Number Invalid";

**return** **NULL**;

**endif**;

}

**public** **function** addBook(Book $book){

$this->books[] = $book;

}

**public** **function** removeBook($number){

**if**(array\_key\_exists($number, $this->books)):

unset($this->books[$number]);

*// $this->books = array\_values($this->books);*

**else**:

**echo** "Book Number Invalid";

**endif**;

}

}

**class** **BookListIterator** **implements** Iterator{

**protected** $library;

**protected** $location = 0;

**public** **function** \_\_construct(Library $library){

$this->library = $library;

}

**public** **function** current(){

**return** $this->library->getBook($this->location);

}

**public** **function** next(){

**if**($this->location < ($this->library->bookCount() - 1)):

$this->location += 1;

**return** $this;

**else**:

$this->location = 0;

**return** $this;

**endif**;

}

**public** **function** prev(){

**if**($this->location == 0):

$this->location = $this->library->bookCount() - 1;

**return** $this;

**else**:

$this->location -= 1;

**return** $this;

**endif**;

}

**public** **function** rewind(){

$this->location = 0;

}

**public** **function** key(){

**return** $this->location;

}

**public** **function** valid($key){

**if**(array\_key\_exists($key, $this->library)):

**return** **TRUE**;

**else**:

**return** **FALSE**;

**endif**;

}

}

*/\*\**

*\* Testing*

*\*/*

$harrypotter = **new** Book("Harry Potter", "J.K Rowling");

$davincicode = **new** Book("Da Vinci Code", "Dan Brown");

$angelsanddemons = **new** Book("Angels and Demons", "Dan Brown");

$outsider = **new** Book("Outsider", "Stephan Hawkings");

$library = **new** Library();

$library->addBook($harrypotter);

$library->addBook($davincicode);

$library->addBook($angelsanddemons);

$library->addBook($outsider);

$iterator = **new** BookListIterator($library);

$iterator->next()->next()->next()->next()->next();

**echo** $iterator->current()->getTitleAuthor()."<br/>";

$iterator->rewind();

**echo** $iterator->current()->getTitleAuthor()."<br/>";

$iterator->prev()->prev();

**echo** $iterator->current()->getTitleAuthor()."<br/>";

?>

**Code 12: iterator.php**

### Composite Pattern

**Definition:** It allows us to compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.

The compoisite pattern allows us to build structures of objects in the form of trees that contain both compoisitons of objects and individual objects as nodes.

Using a compoisite structure, we can apply the same opreations over both composites and individual objects. In other words, in most cases we can ignore the differences between compositions of objects and individual objects

**spl\_object\_hash($object)**

This function returns a unique identifier for the object. This id can be used as a hash key for storing objects or for identifying an object.

**≪Interface≫**

**Graphic**

print()

**CompositeGraphic**

add()

remove()

**Ellipse**

**Fig 11: Composite Pattern UML Diagram**

<?php

**class** **Book**{

**protected** $title;

**protected** $author;

**public** **function** \_\_construct($title, $author){

$this->title = $title;

$this->author = $author;

}

**public** **function** getTitle(){

**return** $this->title;

}

**public** **function** getAuthor(){

**return** $this->author;

}

}

**class** **Library**{

**protected** $books;

**public** **function** add($book){

$this->books[spl\_object\_hash($book)] = $book;

**return** $this;

}

**public** **function** remove($book){

**if**(array\_key\_exists(spl\_object\_hash($book), $this->books)):

unset($this->books[spl\_object\_hash($book)]);

**else**:

**echo** "Sorry this book do not exist";

**endif**;

}

**public** **function** getTitle($book){

**return** $this->books[spl\_object\_hash($book)]->getTitle();

}

**public** **function** getAuthor($book){

**return** $this->books[spl\_object\_hash($book)]->getAuthor();

}

}

$harrypotter = **new** Book("Harry Potter", "J.K Rowling");

$davincicode = **new** Book("Da Vinci Code", "Dan Brown");

$angelsanddemons = **new** Book("Angels and Demons", "Dan Brown");

$outsider = **new** Book("Outsider", "Stephan Hawkings");

$library = **new** Library();

$library->add($harrypotter)->add($davincicode)->add($angelsanddemons)->add($outsider);

**echo** $library->getTitle($harrypotter)." by ".$library->getAuthor($harrypotter)."<br/>";

?>

**Code 12: objectkey.php**

<?php

**interface** Graphic{

**public** **function** print();

}

**class** **CompositeGraphic** **implements** Graphic{

**protected** $graphics;

**public** **function** print(){

**echo** "<pre>";

var\_dump($this->graphics);

**echo** "</pre>";

**foreach** ($this->graphics **as** $graphic):

**echo** get\_class($graphic)."<br/";

**echo** $graphic->print()."<br/>";

**endforeach**;

}

**public** **function** add(Graphic $graphic){

$this->graphics[] = $graphic;

}

**public** **function** remove(Graphic $graphic){

**if**(in\_array($graphic, $this->graphics)):

$key = array\_search($graphic, $this->graphics);

unset($this->graphics[$key]);

$this->graphics = array\_values($this->graphics);

**endif**;

}

}

**class** **Ellipse** **implements** Graphic{

**public** **function** print(){

**echo** ("Ellipse");

}

}

$ellipse1 = **new** Ellipse();

$ellipse2 = **new** Ellipse();

$ellipse3 = **new** Ellipse();

$ellipse4 = **new** Ellipse();

$graphic = **new** CompositeGraphic();

$graphic1 = **new** CompositeGraphic();

$graphic2 = **new** CompositeGraphic();

$graphic1->add($ellipse1);

$graphic1->add($ellipse2);

$graphic1->add($ellipse3);

$graphic2->add($ellipse4);

$graphic->add($graphic1);

$graphic->add($graphic2);

$graphic->print();

?>

**Code 12: composite.php**

### State Pattern

**Definition:** It allows an object to alter its behavior when its internal state changes. The object will appear to change its class.

**Signal**

green

red

state

setState()

nextState()

getGreen()

getRed()

**≪Interface≫**

**State**

on()

**Green**

**Red**

**Fig 11: State Pattern UML Diagram**

<?php

**class** **Signal**{

**protected** $green;

**protected** $red;

**protected** $state = **NULL**;

**public** **function** \_\_construct(){

$this->green = **new** Green();

$this->red = **new** Red();

}

**public** **function** setState(State $state){

$this->state = $state;

}

**public** **function** nextState(){

**return** get\_class($this->state);

}

**public** **function** getGreen(){

**return** $this->green;

}

**public** **function** getRed(){

**return** $this->red;

}

}

**interface** State{

**public** **function** on(Signal $signal);

}

**class** **Green** **implements** State{

**public** **function** on(Signal $signal){

**echo** "Cars, move please!";

$signal->setState($signal->getRed());

}

}

**class** **Red** **implements** State{

**public** **function** on(Signal $signal){

**echo** "Cars, Stop!";

$signal->setState($signal->getGreen());

}

}

$signal = **new** Signal();

$green = **new** Green();

$red = **new** Red();

$green->on($signal);

**echo** $signal->nextState();

$red->on($signal);

**echo** $signal->nextState();

?>

**Code 12: state.php**

### Proxy Pattern

**Definition:** It provides a surrogate or placeholder for another ot control access to it.

Use the Proxy Pattern to create a representative object that controls access to another object, which may be remote, expensive to create or in need of securing.

**Remote Proxy:** The proxy acts as a local representatinve for an object. A method calls on the proxy results in the call being transferred over the wire, invoked remotely, and the result being returned back to the proxy and then to the Client.

**Virtual Proxy:** It acts as a representative for an object that may be expensive to create. Itoften defers the creation of the object until it is needed. It also acts as a surrogate for the object before and while it is being created. After that, the proxy delegates requests directly to the RealSubject.



**Fig 12: Proxy Pattern Diagram**

A proxy is a class functioning as an interface to something else. A proxy implements a subject’s interface. A proxy replaces a subject by Liskov-Substitution Principle

**≪Interface≫**

**Subject**

**SubjectProxy**

**RealSubject**

**Fig 12: Proxy Pattern UML Diagram**

Client consumes Subject ≪interface≫, client consumes RealSubject, client consumes SubjectProxy. SubjectProxy extending RealSubject

**class** **SubjectProxy** **extends** RealSubject{}

**Remote Proxy**



****

**Fig 13: Remote Proxy Diagram**

**Remote Object Example**

<?php

**interface** Image{

**public** **function** display();

}

**class** **RealImage** **implements** Image{

**protected** $filename;

**public** **function** \_\_construct($filename){

$this->filename = $filename;

$this->loadFromDisk($filename);

}

**public** **function** display(){

**echo** "**{**$this->filename**}**";

}

**public** **function** loadFromDisk($filename){

**echo** "Loading: **{**$filename**}**<br/>";

}

}

**class** **ProxyImage** **implements** Image{

**protected** $filename;

**protected** $realImage;

**public** **function** \_\_construct($filename){

$this->filename = $filename;

}

**public** **function** display(){

**if**($this->realImage == **null**){

$this->realImage = **new** RealImage($this->filename);

}

$this->realImage->display();

}

}

$image = **new** ProxyImage("test.jpg");

$image->display();

?>

**Fig 13: proxy.php**

**Remote Object Pros**

* Share Object across multiple systems
* No local memory storage
* Can act as an adapter for a completely different remote object

**Remote Object Pros**

* Complex setup
* Fails on Adapter Failure
* As slow as the protocol

**Firewall Proxy**

It controls access to a set of network resources, protecting the subject from “bad” clients.

**Smart Reference Proxy**

It provides additional actions whenever a subject is referenced, such as counting the number of references to an object.

**Caching Proxy**

It provides temporary temporary storage for results of operations that are expensive. It can also allow multiple clients to share the results to reduce computation or network latency.

**Synchronization Proxy**

It provides safe access to a subject from multiple threads.

**Complexity Hiding Proxy**

It hides the complexity of and controls access to a complex set of classes. This is sometimes called the Façade Proxy for obvious reasons. The Complexity Hiding Proxy differs from the Façade Pattern in that the proxy controls access, while the Façade Pattern just provides an alterantive interface.

**Copy-On-Write Proxy**

It controls the copying of an object by deferring the copying of an object until it is required by a client. This is a variant of the Virtual Proxy

### Compound Pattern

Patterns are often used together and combined within the same design solution. A compound pattern combines two or more patterns into a solution that solves a recurring or general problem.

MVC is a part of compound Pattern

Model: The models holds all the data, state and application logic. The model is oblivious to the view an dcontroller, although it provides on interface to manipulate and retrieve its state and it can send notifications of state changes to observer.

Controller: It takes user input and figures out what it means to the model.

View: It gives us the presentation of the model. The view usually gets the state and data it needs to display directly from the model.



**Fig 14: Model-View-Controller**



**Fig 15: Model-View-Controller**

**View and Controller uses strategy pattern**

**Controller provides the strategy**

**View uses composite pattern, template pattern**

**Model uses the Observer pattern**



**Fig 16: Model-View-Controller**

### Architectural Patterns

They are used to create the living vibrant architecture of buildings, towns, and cities. This is where patterns got their start.

### Applications Patterns

They are patterns for creating system level architecture. Many multi-tier architectures fall into this category.

### Domain-Specific Patterns

They are patterns that concern problems in specific domains, like concurrent sstems or real-time systems

### Business Process Patterns

It describes the interacton between businesses, customers and data, and can be applied to problems such as how to effectively make and communicate decisions

### Organizational Patterns

It describes the structures and practices of human organiations. Most efforts to date have focused on organizations that produce and/ or support software.

### User Interface Design Patterns

It addresses the problems how to design interactive software programs.

### PHP Terms

**array\_search($value, $array)**

It searches an array for a value and returns the key.

**array\_values($array)**

It returns an array containing all the values of an array.

**print\_r($array, true)**

Instead of printing out, it will save to a variable.

**spl\_object\_hash($object)**

This function returns a unique identifier for the object. This id can be used as a hash key for storing objects or for identifying an object.

### Design Principles

* Identify the aspects of our application that vary and separate them from what stays the same.
* Program to an interface, not to an implementation
* Favor composition over inheritance
* Classes should be open for extension, but closed for modification
* Depend upon abstractions. Do not depend upon concrete classes.
* Principle of Least Knowledge – talk only to our immediate friends.
* Don’t call us, we will call you.
* A class should have only one reason to change

### Magic Methods

**\_\_construct()**

**\_\_destruct()**

**\_\_call()**

**\_\_callStatic()**

**\_\_get()**

**\_\_set()**

**\_\_isset()**

**\_\_unset()**

**\_\_sleep()**

**\_\_wakeup()**

**\_\_toString()**

**\_\_invoke()**

**\_\_set\_state()**

**\_\_clone()**

**\_\_debugInfo()**

### Pattern Summary

|  |  |
| --- | --- |
| **Pattern** | **Description** |
| Strategy | Encapsulates interchangeable behaviours and uses delegation to decide which one to use |
| Observer | Allow objects to be notified when state changes |
| Decorator | Wraps an object to provide a new behavior |
| Factory Method | Sublcasses decide which concrete class classes to create |
| Abstract Factory | Allows a client to create families of objects without specifying their concrete classes |
| Command | Encapsulates a request as an object |
| Adapter | Wraps an object and provides a different interface to it |
| Façade | Simplifies the interface of a set of classes |
| Iterator | Provides a way to traverse a collection of objects without exposing its implementation |
| Singleton | Ensures only one and only object is created |
| Template | Subclasses decide which concrete classes to create |
| Composite | Clients treats collections of objects and individual objects uniformly |
| Proxy | Wraps an object to control access to it |
| State | Encapsulates state-based behaviours and uses delegation to switch between behaviours |

|  |  |
| --- | --- |
| **Pattern** | **Examples** |
| Strategy | Custom Superhero |
| Observer | Newletter Subscription |
| Decorator | Coffee + Condiments, Platter + isolation |
| Factory Method | Pizza store of the franchise with different pizza flavor |
| Abstract Factory | Colored Shape Drawing |
| Command | Remote Control |
| Adapter | 110V to 220V Converter |
| Façade | List of things with automation |
| Iterator | Surf through Book Library |
| Singleton | Only One can be created |
| Template | HTML template, different pages different body |
| Composite | One Book and the Library, creating sections and subsections like a tree |
| Proxy | Fake Image creating Real Image by checking Real Image status |
| State | Red and Green Light Switching |

## PHP Object Oriented Solutions

### Scope Resolution

Using the **scope resolution** operator

The scope resolution operator is a pair of colons (∷)

ClassName∷methodorPropertyName

**parent**

This referes to the parent or any ancestor of the current class

parent∷\_\_construct($type,$title)

**self**

This refers to the current class. This is usually used in the class

Instead of using,

**Non-Object Oriented** 🡪 define(“POUNDS\_TO\_KILOGRMS”, 1.234)

**Object-Oriented 🡪** const POUNDS\_TO\_KILOGRAMS = 1.234

**Calling the const, or static** 🡪 self∷POUNDS\_TO\_KILOGRAMS

### Loading classes automatically

**\_\_autoload()**

function \_\_autoload($class){

require\_once(“{$class}.php”);

}

**Files Hierarchy**

 Cost

 Groceries

 Apple.php

 Orange.php

 Autoload.php

 Tools.php

 Transport.php

 ToDoList.php

**Fig 1: Files Hierarchy**

<?php

**class** **ToDoList**{

**public** **function** \_\_construct(){

**echo** "<strong>List of things to do: </strong><br/>";

}

}

?>

**Code 1: ToDoList.php**

<?php

**class** **Tools**{

**public** **function** \_\_construct(){

**echo** nl2br("Tools**\n**");

}

}

?>

**Code 2: Tools.php**

<?php

**class** **Transport**{

**public** **function** \_\_construct(){

**echo** nl2br("Transport**\n**");

}

}

?>

**Code 3: Transport.php**

<?php

**class** **Groceries\_Apple**{

**public** **function** \_\_construct(){

**echo** nl2br("Apple**\n**");

}

}

?>

**Code 4: Apple.php**

<?php

**class** **Groceries\_Orange**{

**public** **function** \_\_construct(){

**echo** nl2br("Orange**\n**");

}

}

?>

**Code 5: Orange.php**

**To get all files inside the path**

<?php

**function** \_\_autoload($class){

$array = explode("\_", $class);

$path = implode(DIRECTORY\_SEPARATOR, $array);

**require\_once**("**{**$path**}**.php");

}

$tools = **new** Tools();

$transport = **new** Transport();

$groceries\_apple = **new** Groceries\_Apple();

$groceries\_apple = **new** Groceries\_Orange();

**Code 6: Autoload.php**

**To get the file outside the path**

**function** \_\_autoload($class){

$array = explode("\_", $class);

$path = **$file\_path**.implode(DIRECTORY\_SEPARATOR, $array);

**require\_once**("**{**$path**}**.php");

}

$todolist = **new** ToDoList();

**Code 7: Autoload(Alternative).php**

### Namespace

#### Using the composer method

🡺 Represents Folder

🡪 Represents File

🡺 src

🡺 Fruits

🡪Apple.php

🡺 Vegetables

🡪Tomato.php

🡪 Inventory.php

🡪 composer.json

**Install composer**

{

"autoload":

{

"psr-4":

{

"Inventory\\" : "src/",

"Fruits\\" : "src/Fruits/",

"Vegetables\\" : "src/Vegetables"

}

}

}

**Code 8: composer.json**

Now type in **composer dump-autoload in the terminal** inside the folder location where composer.json is. This will create a **vendor** folder beside the src folder.

PSR: PHP Standard Recommendation (in this case for autoloading)

PSR-4 is something like 'relative path', PSR-0, 'absolute path'.

**PSR-0 autoload:**

App\Controller\IndexController --> dir/App/Controller/IndexController.php

**PSR-4 autoload:**

App\Controller\IndexController --> dir/IndexController.php

**Relative**: <img src="kitten.png"/>

**Absolute**: <img src="http://www.foo.com/images/kitten.png">

<?php

namespace Fruits;

class Apple

{

public function \_\_construct()

{

echo "Apple"."<br/><br/>";

echo "::::::::::::::::::::"."<br/>";

}

}

?>

**Code 9: Apple.php**

<?php

namespace Vegetables;

class Tomato

{

public function \_\_construct()

{

echo "Tomato"."<br/><br/>";

echo "::::::::::::::::::::::"."<br/>";

}

}

?>

**Code 10: Tomato.php**

<?php

**namespace** Inventory;

**require**("../vendor/autoload.php");

**use** Fruits\Apple;

**use** Vegetables\Tomato;

**class** **Inventory**

{

**public** **function** \_\_construct()

{

$apple = **new** Apple();

$tomato = **new** Tomato();

}

}

$inventory = **new** Inventory();

?>

**Code 11: Tomato.php**

#### Without the composer method

🡺 Represents Folder

🡪 Represents Files

🡺 Vegetables

🡪 Tomato.php

🡺 Fruits

🡪 Apple.php

🡺 Inventory.php

<?php

**namespace** Inventory;

**require** "Fruits/Apple.php";

**require** "Vegetables/Tomato.php";

**class** **Inventory**

{

**public** **function** \_\_construct()

{

$fruits = **new** Apple();

$vegetables = **new** Tomato();

}

}

$inventory = **new** Inventory();

?>

**Code 12: Inventory.php**

<?php

**namespace** Inventory;

**class** **Tomato**

{

**public** **function** \_\_construct()

{

**echo** "Tomato"."<br/><br/>";

**echo** "::::::::::::::::::::"."<br/>";

}

}

?>

**Code 13: Tomato.php**

**Over here we need to use the "namespace Inventory"**

**If we dont use it the the main [Inventory.php] wont recognize this file**

**If the the Inventory.php used "namepace Goods;" then we would have used "namespace Goods"**

<?php

**namespace** Inventory;

**class** **Apple**

{

**public** **function** \_\_construct()

{

**echo** "Apple"."<br/><br/>";

**echo** ":::::::::::::::::::::::"."<br/>";

}

}

?>

**Code 14: Apple.php**

### Using Magic Methods

#### Converting Object to a string

**\_\_toString()**

<?php

**class** **Book**{

**protected** $title;

**protected** $author;

**public** **function** \_\_construct($title, $author){

$this->title = $title;

$this->author = $author;

}

**public** **function** \_\_toString(){

**return** $this->title." by ".$this->author;

}

}

$book = **new** Book("Harry Potter", "J.K Rowling");

**echo** $book;

?>

**Code 15: tostring.php**

#### Cloning an object

**\_\_clone()**

An object’s clone method cannot be called directly.

When we assign an object to another variable, we don’t make a copy of it, instead it creates a reference to the same object. To make a copy of an object, we need to clone it with the **clone** keyword. This creates **shallow copy** of the original object’s properties.

$y = clone $x;

<?php

**class** **Book**{

**protected** $title;

**protected** $author;

**public** **function** setTitleAuthor($title, $author){

$this->title = $title;

$this->author = $author;

}

**public** **function** \_\_toString(){

**return** $this->title." by ".$this->author."<br/>";

}

}

**echo** nl2br("<strong>Creating an Object and echo the class</strong>**\n**");

$book = **new** Book();

$book->setTitleAuthor("Harry Potter", "J.K Rowling");

**echo** $book;

**echo** "<br/>";

**echo** nl2br("<strong>Cloning the object</strong>**\n**");

$copyBook = **clone** $book;

**echo** $copyBook;

**echo** "<br/>";

**echo** nl2br("<strong>New title and author for the clone</strong>**\n**");

$copyBook->setTitleAuthor("Da Vinci Code", "Dan Brown");

**echo** $copyBook;

**echo** $book;

**echo** "<br/>";

**echo** nl2br("<strong>Referring to the same variable using pointer</strong>**\n**");

$a = 3;

$b = &$a;

$b = 5;

**echo** $b."<br/>";

**echo** $a."<br/>";

?>

**Code 16: clone.php**

The above technique works with a class having data members that are of intrinsic type i.e. int, boolean, string, float, However, this technique will not work with a class that has a data member which is an object of another class. In such a scenario, the cloned object continues to share the reference of the data member object of the class that was cloned.

Therefore, we need to define in \_\_clone() method 🡪 For aggregation cases. This is called **deep method**

<?php

**class** **Chess**{}

**class** **Ludu**{}

**class** **playing**{

**public** $game;

**public** **function** \_\_clone(){

$this->game = **clone** $this->game;

}

}

$playing = **new** playing();

$playing->game = **new** Chess();

$clone = **clone** $playing;

$clone->game = **new** Ludu();

**echo** "<pre>";

var\_dump(get\_class($playing->game));

**echo** "</pre>";

**echo** "<pre>";

var\_dump(get\_class($clone->game));

**echo** "</pre>";

?>

**Code 17: clonemethod.php**

#### Accessing properties automatically

A common way of using \_\_get() and \_\_set() is to create protected or private property to store the values of undeclared properties as an associative array. The \_\_get() method checks whether the array element exists; it if does, it returns the value. The set() method simply adds the value to the array.

<?php

**class** **getset**{

**protected** $array;

**public** **function** \_\_get($name){

**if**(array\_key\_exists($name, $this->array)):

**return** $name;

**else**:

**throw** **new** Exception("This value already exist");

**endif**;

}

**public** **function** \_\_set($name, $value){

$this->array[$name] = $value;

}

}

$access = **new** getset();

$access->superman = "superman";

$access->batman = "batman";

**echo** $access->superman."<br/>";

**echo** $access->batman."<br/>";

?>

**Code 18: getset.php**

**\_\_get($one\_argument)**

**\_\_set($first, $second)**

#### Cleaning up

PHP handles memory management through **garbage collection –** the automatic removal of variables and objects when they’re no longer needed by a script.

**\_\_destruct()**

#### Access methods automatically

**\_\_call()** is used to call methods that don’t exist in the current class.

The **\_\_call()** magicmethod takes two arguments: the method name and an array of arguments to be passed to the method.

**call\_user\_func\_array([$object, $functionName],$arguments)**

<?php

**class** **MagicCall**{

**public** **function** \_\_call($functionName, $arguments){

**if**($functionName == "superman"):

$argNum = count($arguments);

**switch**($argNum):

**case** 1:

**echo** "Argument: ".$arguments[0]."<br/>";

**break**;

**case** 2:

**echo** "Argument: **$arguments[0]** **$arguments[1]** <br/>";

**break**;

**default**: **echo** "No more arguments please!";

**endswitch**;

**endif**;

}

}

$magic = **new** MagicCall();

$magic->superman("I can fly so high!");

$magic->superman("I can fly so high!", "My Laser can destroy!");

$magic->superman("I can fly so high!", "My Laser can destroy!", "Boom");

**echo** "<br/>";

**echo** "<br/>";

**class** **MagicCaller**{

**public** **function** \_\_call($functionName, $argument){

**if**($functionName == "batman"):

**return** call\_user\_func\_array([$this, $functionName], $argument);

**endif**;

}

**function** batman($argument){

**echo** "**$argument[0]** **$argument[1]**!<br/>";

}

}

$magical = **new** MagicCaller();

$magical->batman(["I have a batarang!", "Where is my bat mobile?"]);

?>

**Code 19: call.php**

### Exracting Exception Information

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| getMessage() | final | Text of the message passed as the first argument to the exception |
| getCode() | final | Error code passed as second argument |
| getFile() | final | File containing the class definition |
| getLine() | final | Number of the line on which the exception was thrown |
| getTrace() | final | An associative array containing details of the script |
| getTraceString() | final | Same information as getTrace() in the form of string |
| toString() | public | Combines the output of getMessage() and getTraceAsString() |

### Documentation

|  |  |
| --- | --- |
| **Tag** | **Description** |
| @author | Author’s Name |
| @copyright | Copyright information |
| @depracated | Version Infromation |
| @param | Data type, variable name, description |
| @return | Data type, description |
| @var | Data type, description |
| @version | Version details |

Use the dockblockr package for sublime

**Use PHP Documentor**

{

"require-dev": {

"phpdocumentor/phpdocumentor": "^2.0"

}

}

**Code 20: DownloadPHPDocumentor.php**

Go to 🡪 \vendor\bin 🡪 Then type

**phpdoc -d ./src -t ./docs/api 🡪** this will create a doc file inside the docs/api

### Reflection

**Reflection∷export(new ReflectionClass($className))**

Allows us to see the whole structure of the class

<?php

$date = **new** DateTime();

**echo** "<pre>";

var\_dump(**new** ReflectionClass($date));

**echo** "</pre>";

**echo** "<pre>";

Reflection::export(**new** ReflectionClass($date));

**echo** "</pre>";

**echo** "<pre>";

Reflection::export(**new** ReflectionClass('DateTimeZone'));

**echo** "</pre>";

?>

**Code 20: reflection.php**

### Validation

**filter\_has\_var(INPUT\_POST, $var)**

Check whether a varaibles exists in one of the input superglobal arrays, first argument takes the Superglobal constant and the second is the variable.

**filter\_id($filter\_name)**

This returns the numberical value of the PHP constant for a named filter. It takes string as an argument

**filter\_list()**

It returns an array containing the names of supported filters.

**filter\_input(INPUT\_POST, $var, FILTER, OPTIONS)**

It filters to a single variable from a superglobal array. It takes four arguments: superglobal constant, name of the variable, the filter to be applied, any options to be applied to the filter. The last two arguments are optional

**filter\_input\_array(INPUT\_POST, $data\_array, $instructions\_array)**

This filters the contents of a superglobal array according to user-defined criteria. It takes two arguments, superglobal constant and the multidimensional array that defines how each element of the superglobal array is to be filtered.

**filter\_var($var, FILTER, OPTIONS)**

It handles a single variable that comes from an internal source, such as database result or a calculation performed by the script anywhere except the superglobal array. It takes three arguments: name of the variable, the filter to be applied and any options.

**filter\_var\_array($data\_array, $instructions\_array)**

It is identical to filter\_input\_array() except it filters internal data.

**Superglobal constants**

|  |  |
| --- | --- |
| **Constant** | **Superglobal Equivalent** |
| INPUT\_COOKIE | $\_COOKIE |
| INPUT\_ENV | $\_ENV |
| INPUT\_GET | $\_GET |
| INPUT\_POST | $\_POST |
| INPUT\_REQUEST | $\_REQUEST |
| INPUT\_SERVER | $\_SERVER |
| INPUT\_SESSION | $\_SESSION |

#### Thought Process

First var\_dump the filter\_list() then echo out the filter\_id(“int”)

**Setting Filter Options**

|  |  |
| --- | --- |
| **Flag** | **Description** |
| FILTER\_REQUIRE\_ARRAY | Rejects any value that isn’t an array. The filter is applied to all the elements of an array. It also works on multidimensional arrays, applying thhe filter recursively to each level |
| FILTER\_REQUIRE\_SCALAR | Rejects any value that isn’t scalar. In other words, the value must be one of the following data types: Boolean, integer, floating point number, resource (such as a database link or file handle), or string |

#### Validation Filters

**FILTER\_VALIDATE\_BOOLEAN *Flags*: FILTER\_NULL\_ON\_FAILURE**

When used without a flag, returns true for “1”, “true”, “on” and “yes”. When the flag FILTER\_NULL\_ON\_FAILURE is set, returns for “0”, “FALSE”, “off”, “no” and an empty string NULL and for non-Boolean values

**FILTER\_VALIDATE\_EMAIL**

Checks that a value conforms to the email format

**FILTER\_VALIDATE\_FLOAT *Options:* decimal Flags: FILTER\_FLAG\_ALLOW\_THOUSAND**

Checks for a floating-point number or integer; returns false or any other data type. The decimal option permits the use of a comma as the decimal point. Setting the flag accepts numbers containing a thousand, separator (comma is the default, but period is used when decimal is set to “,”) The returned values is always stipped of the thousands separator, with a period as the decimal point.

**FILTER\_VALIDATE\_INT Options: min\_range, max\_range**

**Flags: FILTER\_FLAG\_ALLOW\_OCTAL, FILTER\_FLAG\_ALLOW\_HEX**

Checks for an integer, returns false for any other data type. Specify the minimum and maximum acceptable values as an associative array using min\_range and max\_range. Flags permit octal and hexadecimal numbers. Rejects numbers with a decimal point, even if the fraction is 0, for example: 10.0

**FILTER\_VALIDATE\_IP Flags: FILTER\_FLAG\_IPV6, FILTER\_FLAG\_IPV6**

Flags: FILTER\_FLAG\_NO\_PRIV\_RANGE, FILTER\_FLAG\_NO\_RES\_RANGE

Checks that a value is an IP address. Flags allow us to specify only IPv4 or IPv6 or not from private or reserved ranges.

**FILTER\_VALIDATE\_REGEXP Options: regexp**

Validates a value against a Perl-compatible regular expression. The whole value is returned, not just the part that matches the regular expresson.

**FILTER\_VALIDATE\_URL Flags: FILTER\_FLAG\_SCHEME\_REQUIRED, FILTER\_FLAG\_HOST\_REQUIRED**

**Flags: FILTER\_FLAG\_PATH\_REQUIRED, FILTER\_FLAG\_QUERY\_REQUIRED**

cheks that a value conforms to the format of a URL, optionally with required components as specified by flags.

#### Sanitizing Filters

**FILTER\_SANITIZE\_EMAIL**

Removes all characters except letters, digits and !#$%&;&+-/=?\_{|}~@.[].

**FILTER\_SANITIZE\_ENCODED Flags: FILTER\_FLAG\_STRIP\_LOW, FILTER\_FLAG\_STRIP\_HIGH**

**Flags: FILTER\_FLAG\_ENCODE\_LOW, FILTER\_FLAG\_ENCODe\_HIGH**

URL-encodes a string. Setting the flags optionally stips or encodes characters with an ASCII value of less than 32 (LOW) or greater than 127 (HIGH)

**FILTER\_SANITIZE\_MAGIC\_QUOTES**

Escapes single and double quotes by inserting a backslash in fron to fthem in the same way as the **addslashes()** function

**FILTER\_SANITIZE\_NUMBER\_FLOATFlags: FILTER\_FLAG\_ALLOW\_FRACTION**

**Flags: FILTER\_FLAG\_ALLOW\_THOUSAND, FILTER\_FLAG\_ALLOW\_SCIENTIFIC**

Removes all characters except digits and the plus and minus signs. The flags optionally permit a decimal fraction, the thousands separator, and scientific notation (using uppercase or lowercase E) The decimal point and thousands separator are left untouched. If FILTER\_FLAG\_ALLOW\_FRACTION

is not set, the decimal point is removed but not the fraction, for example, 10.5 becomes 105

**FILTER\_SANITIZE\_NUMBER\_INT**

Removes all characters except digits and the plus and minus signs. The decimal point is removed, if present, but not the fraction. for example, 10.0 becomes 100

**FILTER\_SANITIZE\_SPECIAL\_CHARS Flags: FILTER\_FLAG\_STRIP\_LOW, FILTER\_FLAG\_STRIP\_HIGH**

**Flags: FILTER\_FLAG\_ENCODE\_HIGH**

Converts into HTML entities single and double quotes, >, > & and characters with an ASCII value of less than 32. Setting the flags optionally strips characters with an ASCII value of less than 32 (LOW) ro greater than 127(HIGH), or encodes characters with an ASCII value greater than 127.

**FILTER\_SANITIZE\_STRING Flags: FILTER\_FLAG\_NO\_ENCODE\_QUOTES, FILTER\_FLAG\_STRIP\_LOW**

**Flags: FILTER\_FLAG\_STRIP\_HIGH, FILTER\_FLAG\_ENCODE\_LOW**

**Flags: FILTER\_FLAG\_ENCODE\_HIGH, FILTER\_FLAG\_ENCODE\_AMP**

Strips all tags, including HTML, PHP, and XML. The flags set options to strip or encode special characters with an ASCII value of less than 32 (LOW) or greater than 127 (HIGH), leave quotes unencoded, and encode ampersands (&)

**FILTER\_SANITIZE\_STRING**

This is an alias of **FILTER\_SANITIZE\_STRING**

**FILTER\_SANITIZE\_URL**

Removes all characters except letters, digits, and !#$%&;&+-/=?\_{|}~@.[].

**FILTER\_UNSAFE\_RAW Flags: FILTER\_FLAG\_STRIP\_LOW, FILTER\_FLAG\_STRIP\_HIGH, FILTER\_FLAG\_ENCODE\_LOW**

**Flags: FILTER\_FLAG\_ENCODE\_HIGH, FILTER\_FLAG\_ENCODE\_AMP**

Do nothing, the flags set options to strip or encode special characters with an ASCII value of less than 32 (LOW) or greater than 127(HIGH) and encode ampersands (&)

**FILTER\_CALLBACK**

User Defined function or method

Calls a user-defined function to filter data.

Examples:

**if**(filter\_has\_var(INPUT\_POST, "fullname")){

**echo** nl2br($\_POST['fullname']." Got It!**\n**");

}

**else**{

**echo** nl2br("Couldn't find the required data**\n**");

}

*/\*\**

*\* Checking the filter list*

*\*/*

**echo** "<pre>";

print\_r(filter\_list());

**echo** "</pre>";

*/\*\**

*\* Representing one of th supppoted filters.*

*\*/*

**echo** "<pre>";

**echo** filter\_id('int');

**echo** "</pre>";

*/\*\**

*\* FILTER\_VALIDATE\_INT*

*\*/*

**echo** "<pre>";

$filtered = filter\_input(INPUT\_POST, "age", FILTER\_VALIDATE\_INT);

var\_dump($filtered);

**echo** "</pre>";

*/\*\**

*\* FILTER\_VALIDATE\_FLOAT*

*\*/*

**echo** "<pre>";

$filtered = filter\_input(INPUT\_POST, "age", FILTER\_VALIDATE\_FLOAT, "decimal");

var\_dump($filtered);

**echo** "</pre>";

*/\*\**

*\* Filter using filter\_id*

*\*/*

**echo** "<pre>";

$filtered = filter\_input(INPUT\_POST, "age", filter\_id('float'));

var\_dump($filtered);

**echo** "</pre>";

**Code 21: examples.php**

<!DOCTYPE html>

<html>

<head>

<title>Testing</title>

<meta charset = "UTF-8">

</head>

<body>

<form method = "post" action = "<?php echo **$\_SERVER['PHP\_SELF']**; ?>">

<label **for** = "filtered"> Inputs to be filtered </label><br/>

<?php $save = (isset($\_POST['filtered']))? $\_POST['filtered']: '';?>

<input type = "text" name = "filtered" value = "<?php **echo** $save ?>"/><br/>

<label for = "filters"> Filter </label><br/>

<select name = "filters">

<option value = "int">FILTER\_VALIDATE\_INT</option>

<option value = "boolean">FILTER\_VALIDATE\_BOOLEAN</option>

<option value = "float">FILTER\_VALIDATE\_FLOAT</option>

<option value = "validate\_url">FILTER\_VALIDATE\_URL</option>

<option value = "validate\_email">FILTER\_VALIDATE\_EMAIL</option>

<option value = "validate\_ip">FILTER\_VALIDATE\_IP</option>

<option value = "unsafe\_raw">FILTER\_UNSAFE\_RAW</option>

<option value = "string">FILTER\_SANITIZE\_STRING</option>

<option value = "stripped">FILTER\_SANITIZE\_STRIPPED</option>

<option value = "encoded">FILTER\_SANITIZE\_ENCODED</option>

<option value = "special\_chars">FILTER\_SANITIZE\_SPECIAL\_CHARS</option>

<option value = "email">FILTER\_SANITIZE\_EMAIL</option>

<option value = "url">FILTER\_SANITIZE\_URL</option>

<option value = "number\_int">FILTER\_SANITIZE\_NUMBER\_INT</option>

<option value = "number\_float">FILTER\_SANITIZE\_NUMBER\_FLOAT</option>

<option value = "magic\_quotes">FILTER\_SANITIZE\_MAGIC\_QUOTES</option>

</select><br/>

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

</form>

</body>

</html>

<?php

**if**(isset($\_POST['submit'])):

$filterChoosen = filter\_input(INPUT\_POST, 'filtered',

filter\_id($\_POST['filters']));

**echo** "<pre>";

var\_dump($filterChoosen);

**echo** "</pre>";

**endif**;

?>

**Code 22: test.php**

<?php

*/\*\**

*\* Filter with Flags*

*\* @var float*

*\*/*

$var = 100.98;

$filtered = filter\_var($var, FILTER\_SANITIZE\_NUMBER\_FLOAT, FILTER\_FLAG\_ALLOW\_FRACTION);

**echo** "<pre>";

var\_dump($filtered);

**echo** "</pre>";

*/\*\**

*\* Filter with multiple Flags using pipeline*

*\*/*

$var = "100,000.98";

$filtered = filter\_var($var, FILTER\_SANITIZE\_NUMBER\_FLOAT,

FILTER\_FLAG\_ALLOW\_FRACTION|FILTER\_FLAG\_ALLOW\_THOUSAND);

**echo** "<pre>";

var\_dump($filtered);

**echo** "</pre>";

*/\*\**

*\* Filter with options only*

*\*/*

$var = 2;

$filtered = filter\_var($var, FILTER\_VALIDATE\_INT, ["options" =>

["min\_range" => 5, "max\_range" => 10]

]);

**echo** "<pre>";

var\_dump($filtered);

**echo** "</pre>";

$var = "10,5";

$filtered = filter\_var($var, FILTER\_VALIDATE\_FLOAT, ["options" => ["decimal" => ","]]);

**echo** "<pre>";

var\_dump($filtered);

**echo** "</pre>";

*/\*\**

*\* Filter with options and flags*

*\*/*

$var = "100.789,5";

$filtered = filter\_var($var, FILTER\_VALIDATE\_FLOAT, [

"options" => ["decimal" => ","],

"flags" => FILTER\_FLAG\_ALLOW\_THOUSAND

]);

**echo** "<pre>";

var\_dump($filtered);

**echo** "</pre>";

*/\*\**

*\* Filtering multiple variable*

*\*/*

$data = [

'age' => 21,

'rating' => 4,

'price' => 9.95,

'thousands' => 100,000.95,

'european' => 100.000,95

];

$instructions = [

"age" => FILTER\_VALIDATE\_INT,

"rating" => [

"filter" => FILTER\_VALIDATE\_INT,

"options" => [

"min\_range" => 1,

"max\_range" => 5

]

],

"price" => [

"filter" => FILTER\_SANITIZE\_NUMBER\_FLOAT,

"flags" => FILTER\_FLAG\_ALLOW\_THOUSAND

],

"thousands" => [

"filter" => FILTER\_SANITIZE\_NUMBER\_FLOAT,

"flags" => FILTER\_FLAG\_ALLOW\_FRACTION |

FILTER\_FLAG\_ALLOW\_THOUSAND

],

"european" => [

"filter" => FILTER\_VALIDATE\_FLOAT,

"options" => ["decimal" => ","],

"flags" => FILTER\_FLAG\_ALLOW\_THOUSAND

]

];

$filtered = filter\_var\_array($data, $instructions);

**echo** "<pre>";

var\_dump($filtered);

**echo** "</pre>";

?>

**Code 22: flagoption.php**

**is\_null(var)**

Checks whether the variable is null or not

<?php

**class** **ValidatorSanitizer**{

**protected** $inputType;

**protected** $submitted;

**protected** $required;

**protected** $filterArgs;

**protected** $filtered;

**public** $missing;

**protected** $errors;

**public** **function** \_\_construct($required = **array**(), $inputType = 'post'){

**if**(!is\_null($required) && !is\_array($required)):

**throw** **new** Exception("Required field must be an array");

**endif**;

$this->required = $required;

$this->setInputType($inputType);

**if**($this->required){

$this->checkRequired();

}

$this->filterArgs = **array**();

$this->errors = **array**();

}

**protected** **function** setInputType($type){

**switch**(strtolower($type)):

**case** 'get':

$this->inputType = INPUT\_GET;

$this->submitted = $\_GET;

**break**;

**case** 'post':

$this->inputType = INPUT\_POST;

$this->submitted = $\_POST;

**break**;

**default**: **throw** **new** Exception("Input valid types are get and post");

**endswitch**;

}

**protected** **function** checkRequired(){

$OK = **array**();

**foreach**($this->submitted **as** $name => $value):

$value = is\_array($value)? $value : trim($value);

**if**(!**empty**($value)):

$OK[] = $name;

**endif**;

**endforeach**;

$this->missing = array\_diff($this->required, $OK);

}

**protected** **function** checkDuplicateFilter($fieldName){

**if**(isset($this->filterArgs[$fieldName])){

**throw** **new** Exception("A filter has been already set for this field: **{**$fieldName**}**");

}

}

**public** **function** isInt($fieldName, $min = **null**, $max = **null**){

$this->checkDuplicateFilter($fieldName);

$this->filterArgs[$fieldName] = ["filter" => FILTER\_VALIDATE\_INT];

**if**(is\_int($min)):

$this->filterArgs[$fieldName]['options']['min\_range'] = $min;

**endif**;

**if**(is\_int($max)):

$this->filterArgs[$fieldName]['options']['max\_range'] = $max;

**endif**;

}

**public** **function** isFloat($fieldName, $decimalPoint = ".", $allowThousandSeparator = **true**){

$this->checkDuplicateFilter($fieldName);

**if**($decimalPoint != "." && $decimalPoint != ","):

**throw** **new** Exception("Decimal Point must be a comma or period");

**endif**;

$this->filterArgs[$fieldName] = [

'filter' => FILTER\_VALIDATE\_FLOAT,

'options' => ["decimal" => $decimalPoint]

];

**if**($allowThousandSeparator):

$this->filterArgs[$fieldName]['flags'] = FILTER\_FLAG\_ALLOW\_THOUSAND;

**endif**;

}

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Testing</title>

<meta charset = "UTF-8">

</head>

<body>

<form method = "post" action = "<?php **echo** $\_SERVER['PHP\_SELF']; ?>">

<label for = "fullname"> Name </label><br/>

<input type = "text" name = "fullname"><br/><br/>

<label for = "email"> Email </label><br/>

<input type = "text" name = "email"><br/><br/>

<label for = "description"> Description </label><br/>

<textarea name = "description"></textarea><br/><br/>

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

</form>

</body>

</html>

<?php

**if**(filter\_has\_var(INPUT\_POST, "submit")){

$data = ["fullname", "email", "description"];

$filtered = **new** ValidatorSanitizer($data);

print\_r($filtered->missing);

}

?>

**Code 23: validatorsanitizer.php**

### Versatile Remote File Connector

**ini\_get($var)**

Gets the value of a configuration option

**parse\_url($url)**

Parse a URL and return its components

<?php

$url = 'http://username:password@hostname:9090/path?arg=value#anchor';

var\_dump(parse\_url($url));

var\_dump(parse\_url($url, PHP\_URL\_SCHEME));

var\_dump(parse\_url($url, PHP\_URL\_USER));

var\_dump(parse\_url($url, PHP\_URL\_PASS));

var\_dump(parse\_url($url, PHP\_URL\_HOST));

var\_dump(parse\_url($url, PHP\_URL\_PORT));

var\_dump(parse\_url($url, PHP\_URL\_PATH));

var\_dump(parse\_url($url, PHP\_URL\_QUERY));

var\_dump(parse\_url($url, PHP\_URL\_FRAGMENT));

?>

**Code 24: parseurl.php**

<?php

$url = '//www.example.com/path?googleguy=googley';

var\_dump(parse\_url($url));

?>

**Code 25: parseurlwithoutscheme.php**

**Parse URL Arguments**

|  |  |
| --- | --- |
| scheme | This identifies the types of request, for example http |
| host | Domain name |
| port | Which port it is connected on |
| user | Username for the FTP connection |
| pass | Password for the FTP connecton |
| path | The path to the file |
| query | The query string minus the leading question mark |
| fragment | The fragment identifier at the end of a URL, minus the leading it |

#### Using File Get Contents

**get\_headers($url)**

Fetches all the headers sent by the server in response to a HTTP request

<?php

$url = 'http://www.example.com';

print\_r(get\_headers($url));

print\_r(get\_headers($url, 1));

?>

**Code 25: getheaders.php**

The above example will output something similar to:

Array

(

[0] => HTTP/1.1 200 OK

[1] => Date: Sat, 29 May 2004 12:28:13 GMT

[2] => Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)

[3] => Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

[4] => ETag: "3f80f-1b6-3e1cb03b"

[5] => Accept-Ranges: bytes

[6] => Content-Length: 438

[7] => Connection: close

[8] => Content-Type: text/html

)

Array

(

[0] => HTTP/1.1 200 OK

[Date] => Sat, 29 May 2004 12:28:14 GMT

[Server] => Apache/1.3.27 (Unix) (Red-Hat/Linux)

[Last-Modified] => Wed, 08 Jan 2003 23:11:55 GMT

[ETag] => "3f80f-1b6-3e1cb03b"

[Accept-Ranges] => bytes

[Content-Length] => 438

[Connection] => close

[Content-Type] => text/html

)

*/\*\**

*\* accessDirect Uses file\_get\_contents*

*\* @return null Connects to the remote website*

*\*/*

**public** **function** accessDirect(){

$this->remoteFile = @file\_get\_contents($this->url);

$headers = @get\_headers($this->url, 1);

**if**($headers):

$regex = "/\d{3}/";

preg\_match($regex, $headers[0], $m);

**echo** $m[0];

**endif**;

}

**Code Snippet**

#### Using cURL to retrieve the remote file

1. Intialize a cURL session with the remote server
2. Set options for the way we want to retrieve the remote file
3. Execute the sesson to get the contents of the remote file.
4. Gather information about the session (such as response headers), if required
5. Close the session

**curl\_init($url)**

Initialize a cURL session

**curl\_setopt(curl\_init($url), OPTION, VALUE)**

Set an option for a cURL transfer

|  |  |
| --- | --- |
| Suppress the HTTP headers | curl\_setopt($session, CURLOPT\_HEADER, **FALSE**); |
| Return the remote file as a string | curl\_setopt($session, CURLOPT\_RETURNTRANSFER, **TRUE**); |

**curl\_exec($curl\_session)**

Perform a cURL session

**curl\_getinfo($session, CURLINFO\_HTTP\_CODE)**

Get the http session info (status)

**curl\_close($session)**

Closes the curl session

*/\*\**

*\* useCurl Uses cURL session*

*\* @return null Connects to the remote website*

*\*/*

**public** **function** useCurl(){

**if**($session = curl\_init($this->url)):

curl\_setopt($session, CURLOPT\_HEADER, **FALSE**);

curl\_setopt($session, CURLOPT\_RETURNTRANSFER, **TRUE**);

$this->remoteFile = curl\_exec($session);

**echo** curl\_getinfo($session, CURLINFO\_HTTP\_CODE);

curl\_close($session);

**else**:

$this->error = "Cannot establish cURL session";

**endif**;

}

**Code Snippet**

#### Using Socket Connection

1. Open a socket connection to the remote server
2. Prepare the HTTP headers to request the file from the remote server
3. Sends the headers over the socket connection
4. Capture the response
5. Close the connection
6. Separate the HTTP reponse headers from the body of the file

**fsockopen($hostname, $port, $errno, $errstr, $timeout)**

Open Internet or Unix domain socket connection

#### Remote File Connector

<?php

*/\*\**

*\* @copyright 2018 Photon Enterprise*

*\* @author Shabuktagin Photon Khan khan.photon@gmail.com*

*\* @version 1.0*

*\**

*\*/*

*/\*\**

*\* Remote Connector*

*\* Lets us access with remotely*

*\*/*

**class** **RemoteConnector**{

**protected** $url;

**protected** $remoteFile;

**protected** $error;

**protected** $urlParts;

*/\*\**

*\* Accepts Url (Instantiation)*

*\* @param string $url remote file connectore url*

*\*/*

**public** **function** \_\_construct($url){

$this->url = $url;

$this->checkURL();

**if**(ini\_get('allow\_url\_fopen')):

*// $this->accessDirect();*

*// $this->useCurl();*

$this->useSocket();

**elseif**(function\_exists('curl\_init')):

$this->useCurl();

**else**:

$this->useSocket();

**endif**;

}

*/\*\**

*\* checkURL Checks URL Validity*

*\* @return null*

*\*/*

**public** **function** checkURL(){

$flags = FILTER\_FLAG\_SCHEME\_REQUIRED | FILTER\_FLAG\_HOST\_REQUIRED;

$urlOK = filter\_var($this->url, FILTER\_VALIDATE\_URL, $flags);

$this->urlParts = parse\_url($this->url);

$regex = "/^[^.]+?\.\w{2}/";

$domainOK = preg\_match($regex, $this->urlParts['host']);

**if**(!$urlOK || $this->urlParts['scheme'] != 'http' || !$domainOK):

**throw** **new** Exception("**{**$this->url**}** is not a valid URL");

**endif**;

}

*/\*\**

*\* accessDirect Uses file\_get\_contents*

*\* @return null Connects to the remote website*

*\*/*

**public** **function** accessDirect(){

$this->remoteFile = @file\_get\_contents($this->url);

$headers = @get\_headers($this->url, 1);

**if**($headers):

$regex = "/\d{3}/";

preg\_match($regex, $headers[0], $m);

**echo** $m[0];

**endif**;

}

*/\*\**

*\* useCurl Uses cURL session*

*\* @return null Connects to the remote website*

*\*/*

**public** **function** useCurl(){

**if**($session = curl\_init($this->url)):

*//Suppress the HTTP headers*

curl\_setopt($session, CURLOPT\_HEADER, **FALSE**);

*//Return the remote file as a string*

curl\_setopt($session, CURLOPT\_RETURNTRANSFER, **TRUE**);

*//Get the remote file and store it in the $remoteFile property*

$this->remoteFile = curl\_exec($session);

*//Get the HTTP status*

**echo** curl\_getinfo($session, CURLINFO\_HTTP\_CODE);

*//Close the cURL session*

curl\_close($session);

**else**:

$this->error = "Cannot establish cURL session";

**endif**;

}

*/\*\**

*\* useSocket Uses Socket connection*

*\* @return null Connects to the remote website*

*\*/*

**public** **function** useSocket(){

$port = isset($this->urlParts['port'])? $this->urlParts['port'] : 80;

$remote = fsockopen($this->urlParts['host'], $port, $errno, $errstr, 30);

**if**(!$remote):

$this->remoteFile = **FALSE**;

$this->error = "Couldnt create a socket connection: **$errstr**";

**else**:

*//Add the query string to th epath, if it exists*

**if**(isset($this->urlParts['query'])):

$path = $this->urlParts['path']."?".$this->urlParts['query'];

**else**:

$path = $this->urlParts['path'];

**endif**;

*//Create the request headers*

$out = "GET **$path** HTTP/1.1**\r\n**";

$out .= "Host: **{**$this->urlParts['host']**}\r\n**";

$out .= "Connection: Close **\r\n\r\n**";

*//Send the headers*

fwrite($remote, $out);

*//Capture the response*

$this->remoteFile = stream\_get\_contents($remote);

fclose($remote);

**endif**;

}

**public** **function** \_\_toString(){

**if**(!$this->remoteFile):

$this->remoteFile = '';

**endif**;

**return** $this->remoteFile;

}

}

$url = "http://sphotonkhan.com/";

**try**{

$output = **new** RemoteConnector($url);

**echo** $output;

}

**catch**(Exception $e){

**echo** "Error: ".$e->getMessage();

}

?>

**Code 26: connector.php**

### Looping with SPL

**Use the** Parsing XML Document **library.xml file**

<?php

**echo** nl2br("<strong>Array Iterator</strong>**\n**");

$numbers = [5, 10, 8, 35, 50];

$iterator = **new** ArrayIterator($numbers);

**foreach**($iterator **as** $number){

**echo** $number."<br/>";

}

**echo** "<br/><br/>";

**echo** nl2br("<strong>Limit Iterator</strong>**\n**");

$limiter = **new** LimitIterator($iterator, 0, 3);

**foreach**($limiter **as** $number){

**echo** $number."<br/>";

}

**echo** "<br/><br/>";

**echo** nl2br("<strong>SimpleXML Iterator (file get contents)</strong>**\n**");

$xml = file\_get\_contents("library.xml");

$iterator = **new** SimpleXMLIterator($xml);

**foreach**($iterator **as** $item){

**echo** $item->getName().": ".$item->attributes()."<br/>";

**echo** $item->title."<br/>";

**echo** $item->author."<br/>";

**echo** $item->publisher."<br/><br/>";

}

**echo** "<br/><br/>";

**echo** nl2br("<strong>SimpleXML Iterator (simplexml load file)</strong>**\n**");

$iterator = simplexml\_load\_file("library.xml", 'SimpleXMLIterator');

**foreach**($iterator **as** $item){

**echo** $item->getName().": ".$item->attributes()."<br/>";

**echo** $item->title."<br/>";

**echo** $item->author."<br/>";

**echo** $item->publisher."<br/><br/>";

}

**echo** "<br/><br/>";

**echo** nl2br("<strong>SimpleXML Iterator with Limit Iterator(simplexml load file)</strong>**\n**");

$iterator = simplexml\_load\_file("library.xml", 'SimpleXMLIterator');

$limit = **new** LimitIterator($iterator, 0, 2);

**foreach**($limit **as** $item){

**echo** $item->getName().": ".$item->attributes()."<br/>";

**echo** $item->title."<br/>";

**echo** $item->author."<br/>";

**echo** $item->publisher."<br/><br/>";

}

**echo** "<br/><br/>";

**echo** nl2br("<strong>SimpleXML Iterator with Regex Iterator(simplexml load file)</strong>**\n**");

$iterator = simplexml\_load\_file("library.xml", 'SimpleXMLIterator');

**foreach**($limit **as** $item){

$regex = **new** RegexIterator($item->attributes(), '/03/');

**foreach**($regex **as** $book){

**echo** $book->title."<br/>";

**echo** $item->author."<br/>";

**echo** $item->publisher."<br/><br/>";

}

}

**echo** "<br/><br/>";

**echo** nl2br("<strong>Directory Iterator</strong>**\n**");

$dir = **new** DirectoryIterator(".");

**foreach**($dir **as** $file){

**if**(!$file->isDot() && !$file->isDir()):

**echo** $file."<br/>";

**endif**;

}

**echo** "<br/><br/>";

**echo** nl2br("<strong>Recursive Iterator Iterator with Recursive Directory Iterator</strong>**\n**");

$dir = **new** RecursiveIteratorIterator(**new** RecursiveDirectoryIterator(".."));

**foreach**($dir **as** $file){

**echo** $file."<br/>";

}

**echo** "<br/><br/>";

?>

**Code 27: iterator.php**

#### Filter Iterator Class

FilterIterator class is abstract, so we cant instantiate a FilterIterator object directly. We need to extedn the class by defining, the accept() method, which should return true when the current elements meets whatever conditions we set.

<?php

**class** **PriceFilter** **extends** FilterIterator{

**public** **function** accept(){

$item = $this->current();

**if**($item['price'] <= 40):

**return** **TRUE**;

**endif**;

**return** **FALSE**;

}

}

$books = [

["book" => "Harry Potter", "author" => "J.K Rowling", "price" => 129],

["book" => "Da Vinci Code", "author" => "Dan brown", "price" => 79],

["book" => "Oliver Twist", "author" => "Charles Dickens","price" => 20]

];

$bookIterator = **new** ArrayIterator($books);

$bookIterator = **new** PriceFilter($bookIterator);

**foreach**($bookIterator **as** $key => $value):

print\_r($value);

**endforeach**;

?>

**Code 28: filter.php**

### XML Writer

<?php

$xml = **new** XMLWriter();

$xml->openURI("php://output");

$xml->startDocument("1.0", "UTF-8");

$xml->setIndent(4);

$xml->startElement("library");

$xml->startElement("book");

$xml->writeAttribute("isbn","1340871234");

$xml->writeElement("title", "Harry Potter");

$xml->writeElement("author", "J.K Rowling");

$xml->writeElement("price", "36");

$xml->endElement();

$xml->startElement("book");

$xml->writeAttribute("isbn","1341234234");

$xml->writeElement("title", "Angels & Demons");

$xml->writeElement("author", "Dan Brown");

$xml->writeElement("price", "234");

$xml->endElement();

$xml->endElement();

$xml->endDocument();

$xml->flush();

?>

**Code 28: xmlwriter.php**

### PHP TERMS

**parse\_ini\_file(filename)**

**It parses the ini file**

**filter\_has\_var(INPUT\_POST, $var)**

Check whether a varaibles exists in one of the input superglobal arrays, first argument takes the Superglobal constant and the second is the variable.

**filter\_id($filter\_name)**

This returns the numberical value of the PHP constant for a named filter. It takes string as an argument

**filter\_list()**

It returns an array containing the names of supported filters.

**filter\_input(INPUT\_POST, $var, FILTER, OPTIONS)**

It filters to a single variable from a superglobal array. It takes four arguments: superglobal constant, name of the variable, the filter to be applied, any options to be applied to the filter. The last two arguments are optional

**filter\_input\_array(INPUT\_POST, $data\_array, $instructions\_array)**

This filters the contents of a superglobal array according to user-defined criteria. It takes two arguments, superglobal constant and the multidimensional array that defines how each element of the superglobal array is to be filtered.

**filter\_var($var, FILTER, OPTIONS)**

It handles a single variable that comes from an internal source, such as database result or a calculation performed by the script anywhere except the superglobal array. It takes three arguments: name of the variable, the filter to be applied and any options.

**filter\_var\_array($data\_array, $instructions\_array)**

It is identical to filter\_input\_array() except it filters internal data.

**method\_exists(object, method\_name)**

Checks whether the method exists in the object or not

**addslashes($string)**

Escapes single and double quotes by inserting a backslash in front of them.

**is\_null($var)**

Checks whether the variable is null or not

**ini\_get($var)**

Gets the value of a configuration option

**parse\_url($url)**

Parse a URL and return its components

**get\_headers($url)**

Fetches all the headers sent by the server in response to a HTTP request

**curl\_init($url)**

Initialize a cURL session

**curl\_setopt(curl\_init($url), OPTION, VALUE)**

Set an option for a cURL transfer

**curl\_exec($curl\_session)**

Perform a cURL session

**curl\_getinfo($session, CURLINFO\_HTTP\_CODE)**

Get the http session info

**curl\_close($session)**

Closes the curl session

**fsockopen($hostname, $port, $errno, $errstr, $timeout)**

Open Internet or Unix domain socket connection

**\n**

It was used to move the carriage down

**\r**

It was used to move the carriage back to the left side of the page.

## PHP Tools

### Primitive Types and Checking Function in PHP

|  |  |
| --- | --- |
| **Functions** | **Description** |
| is\_bool() | TRUE or FALSE |
| is\_integer() | A whole number |
| is\_double() | A floating point (number with decimal point) |
| is\_string() | Character data |
| is\_object() | An object |
| is\_array() | An array |
| is\_resource() | A handle for identifying and owkring with external resources such as databases or files |
| is\_null | An unassigned value |

### Pear and Pyrus

**PEAR** (PHP Extension and Application Repository). PEAR is a repository of quality-controlled PHP packages that extend the functionality of PHP. It also a client-server mechanism for distributing and installing packages and for managing interpackage dependencies.

It is collections of packages, organized into broad categories, such as networking, mail, and XML. The pear repository is managed centrally, so that when we use an official PEAR package, we can be sure of its quality.

**Pyrus** (It is the next generation of the PEAR application)

The name cames from the genus of trees and shrubs that includes the pear tree.

### SVN

Apache Subversion is a software versioning and revision control system distributed as open source under the Apache license. Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation. Its goal is to be a mostly compatible successor to the widely used Concurrent Versions System (CVS). It is better to use GitHub

### PHP Unit Testing

**To download PHP unit Testing**

composer require --dev phpunit/phpunit ^7

**To check the version**

➜ ./vendor/bin/phpunit –version

**To run the test**

➜ ./vendor/bin/ phpunit bootstrap="vendor/autoload.php" tests/EmailTest.php

**To run the test with testdox**

phpunit bootstrap="vendor/autoload.php" --testdox tests/EmailTest.php

**src/Email.php**

<?php

**declare**(strict\_types=1);

**final** **class** **Email**

{

**private** $email;

**private** **function** \_\_construct(string $email)

{

$this->ensureIsValidEmail($email);

$this->email = $email;

}

**public** **static** **function** fromString(string $email): self

{

**return** **new** self($email);

}

**public** **function** \_\_toString(): string

{

**return** $this->email;

}

**private** **function** ensureIsValidEmail(string $email): void

{

**if** (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

**throw** **new** InvalidArgumentException(

sprintf(

'"%s" is not a valid email address',

$email

)

);

}

}

}

**tests/EmailTest.php**

<?php

**declare**(strict\_types=1);

**use** PHPUnit\Framework\TestCase;

require\_once("C:\\xampp\htdocs\\PHP\\PHP Unit\\vendor\\bin\\src\\Email.php");

**final** **class** **EmailTest** **extends** TestCase

{

**public** **function** testCanBeCreatedFromValidEmailAddress(): void

{

$this->assertInstanceOf(

Email::class,

Email::fromString('user@example.com')

);

}

**public** **function** testCannotBeCreatedFromInvalidEmailAddress(): void

{

$this->expectException(InvalidArgumentException::class);

Email::fromString('invalid');

}

**public** **function** testCanBeUsedAsString(): void

{

$this->assertEquals(

'user@example.com',

Email::fromString('user@example.com')

);

}

}

#### Test Case Assert Methods

**assertEquals($val1, $val2, $delta, $message)**

Fail if $val1 is not equivalent to $val2. $delta represents an allowable margin of error

**assertFalse($expression, $message)**

Evaluate $expression. Fail if it does not resolve to false

**assertTrue($expression, $message)**

Evaluate $expression. Fail if it does not resolve to true

**assertNotNull($val, $message)**

Fail if $val is null

**assertNull($val, $message)**

Fail if $val is anything othern than null

**assertSame($val1, $val2, $message)**

Fail if $val1 and $val2 are not references to the same object or if they are variables of different types or values.

**assertNotSame($val1, $val2, $message)**

Fail if $val1 and $val2 are references to the same object or variables of the same type and value

**assertRegExp($regexp, $val, $message)**

Fail if $val is not matched by regular expression $regexp

**assertType($typestring, $val, $message)**

Fail if $val is not the type described in $type

**assertAttributeSame($val, $attribute, $classname, $message)**

Fail if $val is not the same type and value as $classname∷$attribute

**fail()**

Fail

#### Some Constrant Methods

**greaterThen($num)**

Test value is greater than $num

**contains($val)**

Test value (traversables) contains an element that matches $val.

**identicalTo($val)**

Test value is a reference to the same object as $val or, for non-objects, is of the same type and value.

**greaterThanOrEqual($num)**

Test value is greater than or equal to $num

**lessThan($num)**

Test value is less than $num

**lessThanOrEqual($num)**

Test value is less than or equal to $num

**equalTo($value, $delta=0, $depth=10)**

Test value equals $val. If specified, $delta defines a margin or error for numeric comparisons, and $depth determines how recursive a comparison should be for array or objects.

**stringContains($str, $cassensitive = true)**

Test value contains $str. This is case sensitive by default.

**matchesRegularExpression($pattern)**

Test value matches the regular expression in $pattern

**logicalAnd( PHPUnit\_Framework\_Constraint $const, [,$const..])**

All provided constraints pass

**logicalOr( PHPUnit\_Framework\_Constraint $const, [,$const..])**

At least one of the provided constaints match

**locialNot(PHPUnit\_Framework\_Constraint $const)**

The provided constraint does not pass

#### Some Matcher Methods

**any()**

Zero or more calls are made to corresponding method (useful for stub objects that return values that don’t test invocations)

**never()**

No calls are made to corresponding method

**atLeastOnce()**

One or more calls are made to corresponding method.

**once()**

A single call is made to corresponding method

**exactly($num)**

$num calls are made to corresponding method

**at($num)**

A call to corresponding method made at $num index (each method call to a mock is recorded and indexed).

### Selenium

Selenium is a portable software-testing framework for web applications. Selenium provides a playback (formerly also recording) tool for authoring tests without the need to learn a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) to write tests in a number of popular programming languages, including C#, Groovy, Java, Perl, PHP, Python, Ruby and Scala.