Introduction to Web Tech

Global Variables

Global Variables	Description
$_COOKIE	Contains any cookie value passed as a part of the request. Key is name of the cookie.
$_GET	Contains GET request parameters. Key is names of form parameters.
$_POST	Contains POST request parameters. Key is names of form parameters.
$_FILES	Contains uploaded files info.
$_SERVER	Contains web server info.
$_ENV	Contains environment variable values. Key is name of the environment variable.

GET vs POST

- **GET Method:**
  - This method encodes form data in URL called **query string**.
  - Visible in browser address window.
  - Result can be bookmarked.

  ☐ See programming examples.
- **POST Method:**
  - This method encodes form data in the body of HTTP request.
  - Data is not shown in URL.
  - Data can’t be bookmarked.

  ☐ See programming examples.

Processing Forms

- Automatic Quoting of Parameters
  - **Magic Quotes** is the name of a PHP feature that automatically quotes input data.

    - When on, all ‘, “, \ and NULL characters are escaped with a backslash automatically.
    - For example, “Oh’ My Dear” is submitted as “Oh\’ My Dear”.

  ☐ See programming examples.

- Self Processing Pages
    - **PHP_SELF** is used in $_SERVER to self process a page.

  ☐ See programming examples.

- Sticky Forms
    - Result of a query is accompanied by a search forms whose default values are those of the previous queries.

  ☐ See programming examples.

- Multivalued Parameters
    - **Select tag** in HTML is used to allow multiple selections. Name field in this tag must be ended with [].

  ☐ See programming examples.

- File Uploads
    - $FILES[’example_file_name’] contains this parameters:

Key	Value	Description
Name	String(8)	Name of the file.
Type	String(10)	MIME type of the file.
Tmp_name	String(14)	Temporary file name on the server system.
Error	int(0)	Error code.
Size	int(2045)	Size of the file.

    - **is_uploaded_file()** is used to check if file is successfully uploaded or not.

  ☐ See programming examples.

- Setting Header Response
    - Before any of the body is generated the header() function should be called. i.e. header() must be on the top even before <html> tag.

Maintaining State

- **Cookies:**
  - A cookie is a string that contains several fields. Server can send one or more cookies to a browser in the header response.
  - Syntax: *setcookie(name, [,value [,expire [,path [,domain [,secure]]]]]);*

Name	Description
Name	Unique name for a cookie. Name must not contain whitespace or semicolon.
Value	Arbitrary string value attached to a cookie.
Expire	Specified from midnight Jan 1 1970 GMT. eg, time() + 60*60 for cookie to be expired in 1 hour. If nothing is set it will expire on browser exist.
Path	Defines the directory in which current page resides.
Domain	Browser will return the cookie only for URLs within this domain. Default is server hostname.
Secure	Browser will transmit the cookie only over secure connection. Default is false.

  - We can access cookie through $_COOKIE array.

  ☐ See programming examples.

- **Sessions:**
  - It allows to create multipage forms, save user authentication info from page to page and store persistent user preferences on a site.
  - **session_start()** starts the session before document has been generated. It loads registered variable in array $HTTP_SESSION_VARS.
  - **session_register()** register a variable into a session and **session_unregister()** unregisters a variable.
  - **session_destroy()** ends the session.

  ☐ See programming examples.

PHP Error Handling

- Using die() method:
  - This method print a message and exit from current code.

  ☐ See programming examples.
- Custom Error Handling:
  - Syntax: *error_function($error_level, $error_message, $error_file, $error_line, $error_context)*

  ☐ See programming examples.
- Exception Handling:
  - **Try**: Code to be monitored is written here.
  - **Catch**: This block retrieves an exception and creates an object.
  - **Throw**: This triggers exception. Each “Throw” must have at least one catch block.

  ☐ See programming examples.

💡 See solved examples on page no. 1-31 and 1-32

XML

- XML: Extensible Markup Language
- Features of XML:
    1. Separates data from HTML.
    2. Simplifies data sharing.
    3. Simplifies data transport.
    4. Simplifies platform changes.
    5. Makes data more available.
- Advantages of XML:
    1. Allows data interchange between computer systems.
    2. Provides domain specific vocabulary.
    3. It’s about carrying information.
    4. Allows granular updates.
    5. Enable smart searches.
- Disadvantages of XML:
    1. Lack of application processing.
    2. Repetition of elements.
    3. External references.
- Uses of XML:
    1. Web publishing.
    2. Web searching.
    3. Data transfer.
    4. Creates other languages.

- XML Document Structure:
    - XML document must include XML version and file encoding.

    - Valid XML document contain a Document Type Definition(DTD) or an XML schema.
    - XML doc contains one or more elements, each of which may contains more attributes.
    - Elements can contain one or more elements.
    - Attributes contains extra info about specific tag.
- Well Formed XML Document:
    - Only one parent element. i.e. XML doc has only one root element.
    - Begins with XML declaration of version and encoding.
    - Case sensitive.
    - If the XML declaration appears, it must be the very first thing in the doc.
    - An element consists of a start tag (”<”), possibly followed by text and other complete elements and then followed by an end tag (”>”).
    - Elements may not overlap, they must be properly nested.
    - XML has a special empty tag that represents both the start and end tag. eg. <hr/> is same as <hr></hr>.
    - Has two kind of references:
        - **Entity References:** An entity reference like “&”, contains a name (in this case, “amp”) between the start and end delimiters.
        - **Character References:** A character reference, like “&”, contains a hash mark (”#”).

- XML Parser:
    - The parser converts XML into a JavaScript accessible object.
    1. ***xml_parser_create(encoding)***: Function creates an XML parser.
    2. ***xml_parser_into_struct(parser, xml, value_arr, index_arr)***: Function parses XML data into an array.
- XML Document Object Model:
    - The XML DOM defines a standard way for accessing and manipulating XML documents.

1. **domxml_new_doc():** Creates new XML document.
2. **domxml_open_file():** Opens an XML document file as a DOM object.
3. **domxml_open_mem():** Create a DOM object from an XML doc already in memory.
4. **create_element():** Creates a new element.
5. **append_child():** Appends an element as a child of an existing element.
6. **DomDocument → get_element_by_tagname:** Returns array with nodes with given tagname in document or empty array if not found.

- SimpleXML:
  - SimpleXML is an extension of PHP. This extension has a set of functions that are used to handle the XML and to convert them into an object.
  - It converts XML doc into object like this:
    1. Elements: Are converted to single attributes of the SimpleXML element object.
    2. Attributes: Are accessed using associative arrays.
    3. Element Data: Text data from elements are converted to strings.
  - SimpleXML performs these tasks easily:
    1. Reading XML files.
    2. Extracting data from XML strings.
    3. Editing text nodes or attributes.
- SimpleXML Extensions:
  - Four Simple rules are:
    1. Properties denote element interators.
    2. Numeric indices denote elements.
    3. Non-numeric indices denote attributes.
    4. String conversion allows access to TEXT data.
- SimpleXML functions:
  1. **simplexml_load_file**: Converts XML doc to an object of class SimpleXMLElement.

2. **simplexml_load_string**: Takes an xml string and returns an object of class SimpleXMLElement.
3. **simplexml_import_dom**: Takes a node of DOM doc and makes it into a simpleXML node.
4. **SimpleElement → xpath**: This method searches the SimpleXML node for children matching the Xpath path.
5. **simplexml_element → children**: This method finds children of the element of which it is a member.
6. **simplexml_element → attributes**: This function provides the attributes and values defined within an xml tag.
7. **SimplexmlElement → asxml**: This method formats the parent object's data in XML version 1.0.

XML Schema(XSD)	Document Type Definition(DTD)
Contains namespace awareness. It gives element or attribute into context.	Lacks namespace awareness.
Implements strong typing.	Doesn't implements strong typing and has no way of validating the content to data types.

💡 See solved examples from page no. 2-25 and onwards.

JavaScript and JQuery

- **DHTML**
  - Dynamic HyperText Markup Language.
  - It makes it possible for web pages to interact and change in response to the users actions.
  - DHTML is a combination of JavaScript, CSS and DOM.
- **What is JavaScript?**
  - Designed to add interactivity to HTML pages.
  - It is a scripting language.
  - It is embedded directly into HTML pages.
  - It is an interpreted language.
  - It is case sensitive.

Java	JavaScript
OOP programming language.	OOP scripting language.
Creates application that runs on VM or browser.	Runs only on browser.
Needs to be compiled.	Code are all in text.

- **What can JS do?**
  1. Gives HTML designers a programming tool.
  2. Can put dynamic text into an HTML page.
  3. Can read and write HTML elements.
  4. Can be used to validate data.
  5. Can be used to detect the visitor’s browser.
  6. Can be used to create cookies.
- **Data Types in JS**
  1. **Number**
  2. **String**
  3. **Boolean**

4. **Undefined**: It is the value of a variable with no value.
5. **Null**: A special keyword for null value.
   - var keyword is used for declaring variables in JS.
   - Number, Boolean and String have wrapper classes.

☐ See programming examples.

- Implicit Type Conversion
  1. Catenation (+) coerces numbers to string.
  2. Numeric operators (other than +) coerce strings to numbers.
- Explicit Type Conversion
  1. Use of String and Number constructors.

     eg. *var str_value = String(value);*

     ☐ See programming examples.
  2. Use **toString** method.

     ☐ See programming examples.
  3. Use **parseInt** and **parseFloat** on strings.

     ☐ See programming examples.
- Some date objects

  ☐ See programming examples.
  1. **getDate**
  2. **getMonth**
  3. **getFullYear**
  4. **getTime**
  5. **getHours**
- JS HTML DOM Events
  - Events are normally used in combination with function, and the function will not be executed before the event occurs (such as when a user clicks a button).

Event	Description

Event	Description
onclick	When the user clicks on an element.
onmouseup	When the user releases a mouse button over an element.
onmousedown	When the user presses a mouse button over an element.
onmouseover	When the pointer is moved onto an element, or onto one of its children.
onmouseout	When the user moves the mouse pointer out of an element, or out of one of its children.
onload	When user enters the page.
onunload	When user leaves the page.

☐ See programming examples.

- String Properties

Property	Description
Constructor	Returns the function that created the string object's prototype.
Length	Returns the length of a string
Prototype	Allows you to add properties and methods to an object.

☐ See programming examples.

Method	Description
charAt()	Returns the character at specified index.
concat()	Joins two or more strings and returns a copy of the joined string.
indexOf()	Returns the index of first found occurrence of a specified value in a string.
replace()	Searches a string for the value and returns a new string with replaced value.
search()	Searches a string for a value and returns the position of the match.
slice()	Extracts a part of a string and returns a new string.
substr()	Extracts part of string from starting to a number of index.
substring()	Extracts parts of string between two specified indexes.

☐ See programming examples.

- JS Popup Boxes

1. **Alert Box**: An alert box is often used if we want to make sure information comes through to the user. User will receive “OK” to proceed. Syntax : *alert(”sometext”);*

   ☐ See programming examples.

2. **Confirm Box**: Used to verify something. User will receive “OK” and “Cancel”. Syntax : *confirm(”sometext”);*

   ☐ See programming examples.

3. **Prompt Box**: If we want user to input some text before entering next page. User will receive “OK” and “Cancel”. Syntax : *prompt(”sometext”,”defaultvalue”);*

   ☐ See programming examples.

- **JQuery:** It is a JavaScript library designed to simplify and standardize interactions between JS code and HTML elements.
- Features of JQuery:
  1. HTML/DOM manipulation.
  2. CSS manipulation.
  3. HTML event methods.
  4. Effects and animations.
  5. AJAX
  6. Utilities
- JQuery selectors:
  1. #id selector
     - This selector uses id attributes of an HTML tag to find the specific element.
     - Id should be unique within a page.
     - Used to find single, unique element.

     ☐ See programming examples.

  2. .class selctor
     - This selector finds elements with a specific class.

☐ See programming examples.

- DOM Manipulation methods:

Method	Description
after(content)	Insert content after each of the matched element.
append(content)	Append content after to the inside of every matched element.
appendTo(content)	Append all of the matched elements to another, specified, set of elements.
before(content)	Insert content before each of the matched elements.
empty()	Remove all child nodes from the set of the matched elements.
replaceWith(content)	Replace a complete DOM element with a specified HTML or DOM element.
remove(expr)	Removes all matched elements from the DOM.

☐ See programming examples.

CodeIgniter

- **CodeIgniter** is an application development framework. It is very popularly used to develop websites, using PHP. It is an open source framework.
- CodeIgniter Features:
  1. Model-View-Controller(MVC) based system.
  2. Extremely light weight.
  3. Fast and has better performance over other framework.
  4. Full featured database classes with support for several platforms.
  5. Query builder database support.
- Application Architecture MVC Framework:
  - CodeIgniter is based on the MVC development architecture.
  - MVC architecture separates application logic from presentation.
  - Using this architecture, it keeps the data, the presentation and flow through the application as separate parts.
    i. The **Model** represents data structures. The model classes contain functions as handle and manipulate information in database.
    ii. The **View** is the information that is presented to a user. A view will normally be a web page but in CodeIgniter, a view can also be a page fragment like a header or footer. Views get the data to displays from the controllers.
    iii. The **Controller** serves as an intermediary between the Model, the view and any other resources needed to process the HTTP request and generate a web page. Thus, the controller handles HTTP requests - redirects, authentication, web safety, encoding, etc.

    Controllers handle input collected from the user and then pass it to the model for further processing. A controller is actually a class file that is named to associate with a URI.

- Libraries:
  - Library can be loaded as: *$this→load→library('class name');*

- Multiple libraries can be loaded as: *$this→load→library (array('email', 'table'));*

Library class	Description
Input class	This class pre-processes the input data for security reason.
Output class	This class sends the output to browser and also caches that webpage.
File Uploading class	Provides functionalities related to file uploading.
Form Validation class	Provides various functions to validate form.
Loader class	Loads elements like View files, Drivers, Helpers, Models, etc.
Encryption class	Provides two-way data encryption functionality.
Migration class	Provides functionality related to database migrations.

- Working with Database:
  - Let us consider a table Student(Roll, Name).

  1. **Connecting to a database**:
     a. *$this→load→database();*
  2. **Inserting a record**:
     a. $data = array('Roll' ⇒ '101', 'Name' ⇒ 'ABC');
     b. *$this→db→insert("Student", "$data);*
  3. **Update a record**:
     a. $data = array(….
     b. *$this→db→set($data);*

        $this→db→where("Roll", '101');

        $this→db→set("Student", $data);
  4. **Delete a record**:
     a. *$this→db→delete("Student", "Roll = 101");*
  5. **Close a connection**:
     a. *$this→db→close();*

- More functions to handle database:

1. Query() : Instead of using separate functions for different query types. The query() function can be used.
2. getResultArray() : To get the result in array.
3. getRow() : Returns first row of the result.
4. getRowArray() : Returns the array of the rows of the result.
5. getNumRows() : Returns the number of records returned by the query.
6. getFieldCount() : Returns the number of fields returned by the query.
7. getFieldNames() : Returns an array with the names of the fields returned by the query.

- Page Redirection:
  - **redirect()** : *redirect($uri = ‘ ‘, $method = ‘auto’, $code = NULL)*
  - Parameters:
    1. $uri(string) : URI string.
    2. $method(string) : Redirects method.
    3. $code(string) : HTTP response code.
- Add Session Data:
  - To initialize a session this constructor is used :
    $this→load→library(’session’);
  - **set_userdata()** : *$this→session→set_userdata(’name’, ‘value’);*
  - Parameters:
    1. name is the name of the session variables.
    2. value is the value of the stored.
- Fetch Session Data:
  - **userdata()** : *$name = $this→session→userdata(’name’);*
- Remove Session Data:
  - **unset_userdata()** : *$this→session→unset_userdata(’name’);*
- Set Cookie Data:

- **set_cookie()** : *set_cookie($name[, $value = ‘ ‘[, $expire = ‘ ‘[, $domain = ‘ ‘[, $path = ‘/’[, $prefix = ‘ ‘[, $secure = FALSE[, $httponly = FALSE]]]]]]])*

- Fetch Cookie Data:
  - **get_cookie()** : *get_cookie($index[, $xss_clean = NULL])*
  - Parameters:
    1. $index(string) : Cookie name.
    2. $xss_clean(bool) : Whether to apply XSS filtering to the returned value.
- Remove Cookie Data:
  - **delete_cookie()** : *delete_cookie($name [, $domain = ‘ ‘[, $path = ‘/’[, $prefix = ‘ ‘]]]]])*
  - Parameters:
    1. $name(string) : Cookie name.
    2. $domain(string) : Cookie domain.
    3. $path(string) : Cookie path.
    4. $prefix(string) : Cookie name prefix.