Web Engineering Summary

Lecture 1:

- The World Wide Web ("WWW" or simply the "Web") is a system of interlinked, hypertext documents that runs over the Internet.
- WWW: Basic Ideas
  - Hypertext/hyperlink
  - Resource Identifiers
    - Unique identifiers used to locate a particular resource.
    - **URI** (Uniform Resource Identifier)/**URL** (Uniform Resource Locator)
  - Markup language: characters or codes embedded in text which indicate.
- Web 1.0 → Syntactic/structural web
- Web 2.0 → Social web
  - The Web where "ordinary" users can meet, collaborate, and share using social software applications on the Web.
- Web 3.0 → Semantic web
  - "An extension of the current Web in which information is given well defined meaning, better enabling computers and people to work in cooperation."
  - Semantic web is for integrating and searching.
- Ontology provides a **shared and common** understanding of a domain that can be communicated across people and application systems.

Web 1.0 / 2.0 / 3.0 Summary

Crawl	Walk	Run
Web 1.0	**Web 2.0**	**Web 3.0**
Mostly Read-Only	Wildly Read-Write	Portable & Personal
Company Focus	Community Focus	Individual Focus
Home Pages	Blogs / Wikis	Lifestreams / Waves
Owning Content	Sharing Content	Consolidating Content
Web Forms	Web Applications	Smart Applications
Directories	Tagging	User Behavior
Page Views	Cost Per Click	User Engagement
Banner Advertising	Interactive Advertising	Behavioral Advertising
Britannica Online	Wikipedia	The Semantic Web
HTML / Portals	XML / RSS	RDF / RDFS / OWL

Lecture 2:

- Static Websites:
    - Any site that has **fixed content** usually written in html code.
    - Every page will have the code written separately.
    - Every page has to be saved separately on the server.
    - Changes have to be made **manually** every time, and you need coding knowledge to make any and all changes.
    - Static Web Page Files are published by **physical transfer** from the development PC to a Web Hosting Computer.

Advantages	**Disadvantages**
Easy to develop.	Requires web development expertise to update site.
Cheap to develop.	Changes and updates are very time consuming.
Cheap to host.	Site not as useful for the user.
	Content can get stagnant.
	Out of date.

- Dynamic Websites:
    - A site whose construction is controlled by an **application server** processed by **server-side** scripts.
    - Pages of the website are not coded and saved separately.
    - The design/template is saved separately.
    - Corresponding contents are saved separately.
    - The pages are dynamically populated every time.
- Web Content Management System (WCMS) is a Content Management Systems (CMS) designed to simplify the publication of Web content to Web sites, in particular allowing content creators to submit content without requiring technical knowledge of HTML or the uploading of files. [The drawback of this is SECURITY]

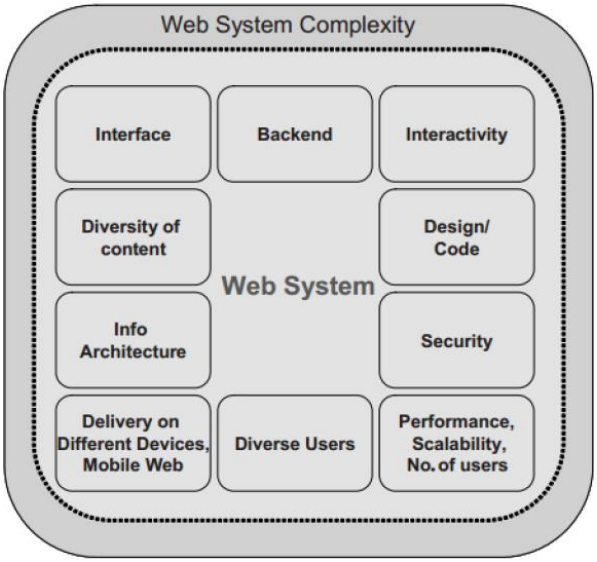
Advantages	**Disadvantages**
Much more functional website.	More expensive to develop.
Much easier to update.	Slower to develop.
New content brings people back to the site and helps in the search engines.	Hosting costs a little more.
Can work as a system to allow staff or users to collaborate.	

- Is static right for your organization?
    - Limited Budget.
    - 20 pages or less.
    - Add new pages occasionally.
    - Skilled Staff.
    - Portal and physical web pages files.

- Is dynamic right for your organization?
    - Need custom programming.
    - Need pages that must pull information.
    - Need to easily add and edit web pages.
    - Need pages that display and update records.
    - Need to give one or more staff member the access to edit.
    - Need convenient access to setup common interactive features.
- Browsers:
    - Browsers are the interpreters of the web.
    - They request information and then receive it.
- HTML = Hypertext Markup Language
- CSS = Cascading Style Sheet.
- Frameworks:
    - Frameworks are built to make building and working with programming languages easier.
    - Take all the difficult, repetitive tasks in setting up a new web application and either do them for you or make them very easy to do.
- Databases:
    - **SQL** provides more structure which helps with making sure all the data is correct and validated.
    - **NoSQL** provides a lot of flexibility for building and maintaining applications.
- There are typically multiple clients interacting with the same application stored on a server.
- The server will gather the appropriate information and respond to those requests.
- API is created by the developer of an application to allow other developers to use some of the applications functionality without sharing code.
- Data formats are the structure of how data is stored.

Lecture 3:

- Web Engineering is a science that extends Software Engineering to Web applications, but with Web centric approaches.
  - Control.
  - Risk minimization.
  - Enhanced maintainability and quality.
- Unlike traditional software, the Web serves as both **development** & **user** platform.
- A Web application is a system that utilizes standards technologies to deliver Web specific resources to clients (typically) through a browser.
- An appropriate infrastructure is necessary to support the growth of a Web based system in a flexible and controlled manner.



- Application development on the Web remains largely ad hoc
  - Individual experience
  - Little or no documentation for code/design
- Short term savings lead to long term problems in operation, maintenance, usability.
- Top Web project pitfalls:
  - Not meeting functionality and the users' needs.
  - Security breaches.
  - Including errors and crashes.
  - Poor usability/performance/maintainability/scalability/project management.
- A well-engineered Web system:
  - is scalable/portable/reusable/maintainable/secure/usable/robust and reliable.
  - is functionally complete and correct.
  - performs reasonably even under flash and peak loads.
  - is interoperable with other systems.
  - has universal accessibility.
  - is well documented.
- Portal-Oriented Web Applications:
  - Business portals.
  - Marketplace portals.
  - Community portals.

- Characteristics of Web Apps:
    - Product (The building blocks of a Web application):
        - Content
        - Navigation Structure (Hypertext)
        - User interface (presentation)
    - Usage:
        - Social context (users)
        - Technical context (network & devices)
        - Natural context (place & time)
    - Development:
        - The development team
        - Technical infrastructure
        - Process
        - Integration
    - Evolution
        - Continuous change
        - Competitive pressure
        - Fast pace
- Key knowledge areas:
    - Software Engineering
    - Network Engineering
    - Hypermedia
    - Information systems

Lecture 4:

- Requirements Engineering (RE): the principles, methods, & tools for eliciting, describing, validating, and managing project goals and needs.
- What are the consequences?
    - Inadequate software architectures.
    - Budget overruns.
    - Production delays.
    - Low user acceptance.
- A requirement describes a property to be met or a service to be provided by a system.

- Types of requirements:
  - Functional - describes the capability's purpose.
    - Data
    - Interface
    - Navigational
    - Personalization
    - Transactional
  - Non-functional - describes the capability's properties.
    - Content
    - Quality
    - System environment
    - User interface
    - Evolution
    - Project constraints
- Top 6 distinguishing characteristics of RE of Web:
  - Multidisciplinary teams.
  - Unavailability of stakeholders.
  - Rapidly changing requirements & constraints.
  - Unpredictable operational environment.
  - No manual for the user interface.
  - Content Management.
- The requirements collection process:
  - Elicitation → specification → validation & verification → management
- Elicitation & Negotiation (learning process):
  - Stakeholders → those that directly influence the requirements.
  - Techniques:
    - Interviewing
    - Joint application design
    - Brainstorming
    - Concept mapping
    - Storyboard
    - Use case modeling
    - Questionnaires
  - Challenges with stakeholders:
    - Users don t know what they want.
    - Lack of commitment.
    - Ever expanding requirements.
    - Communication delays.
    - Users don't take part in reviews.
    - Users don't understand the technology/process.

    - Challenges with developers:
        - Users and engineers/developers speak different “languages”.
        - The tendency to “shoehorn” the requirements into an existing model
            - Saves time for developers, but results may not meet user’s needs.
        - Lake of negotiating skills and domain knowledge.
- Specification:
    - Traditional RE:
        - Stories
            - Plain language scenarios; understandable to non-technical persons.
        - Itemized requirements
            - Plain language lists of requirements.
        - Formatted requirements
            - Accurately defined but allow for plain language descriptions.
        - Formal specifications
            - Expressed in formal syntax & semantics.
    - RE for web apps:
        - Formatted requirements (i.e., use cases) and stories are heavily used.
        - Low to medium accuracy is suitable for Web apps, formal specifications very rarely required.
        - Keep effort for eliciting and managing requirements low.
        - Scalability is important.
- Validation and management:
    - Validation techniques:
        - Review or walk-through
            - Reading and correcting the requirements definition documentation and models.
        - Audit
            - Partial check of the results presented in the review documentation.
        - Traceability matrix
            - Comparison of the application objectives with the requirements of the system.
        - Prototyping for validation
            - Implement a partial set of functional requirements but provide a global vision of the user interface.

Lecture 5

- Why should we create models?
  - Define an abstract view of a real-world entity.
  - Tools of thought.
  - Means of communication.
- Cost of diagrams is way cheaper than cost of code.
- The Unified Modeling Language (UML) is a visual language for specifying and documenting the artifacts of systems.
  - Structural – class diagrams.
  - Behavioral – use cases, sequence diagrams and state machine diagrams.
- Content model (see figure in lecture):
  - Purpose - to model the information requirements of a Web application.
- Hypertext Modeling (see figure in lecture):
  - Purpose - To model the navigation paths available to users.
  - Hypertext Structure Model - navigating among classes.
  - Use "<<navigation class>>" annotation to distinguish from content classes.
  - HDM = Hypertext Design Model
  - WebML = Web Modeling Language
- Presentation Modeling (see figure in lecture):
  - Purpose - To model the look feel of the Web application at the page level.
  - Levels:
    - Page - "root" element; equivalent to a page container.
    - Unit - A fragment of the page logically defined by grouping related elements.
    - Element - text, images, buttons, fields.

PHP (reference : https://www.w3schools.com):

- A PHP script starts with **<?php** and ends with **?>**.
- A PHP file normally contains HTML tags, and some PHP scripting code.
    - If you want to write HTML, it should be inside **echo** or **outside the php tags**.
- PHP statements end with a semicolon (**;**).
- In PHP, keywords (e.g., if, else, while, echo, etc.), classes, functions, and user-defined functions are **not case-sensitive**. However, all variable names are **case-sensitive**!
- In PHP, a variable starts with the **$** sign, followed by the name of the variable.
- We do not have to tell PHP which **data type** the variable is.
- A variable name must start with a **letter** or the **underscore character**.
- A variable name cannot start with a **number**.

```php
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

- The following examples will produce the same output:

```php
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>

<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!"; // the . is for concatenation
?>
```

- The following example will output the sum of two variables:

```php
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

- With PHP, there are two basic ways to get output: echo and print. They are more or less the same. They are both used to output data to the screen.
- A string can be any text inside quotes. You can use single or double quotes.

PHP Arithmetic Operators			
Operator	**Name**	**Example**	**Result**
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y power

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

PHP Comparison Operators			
Operator	**Name**	**Example**	**Result**
==	Equal	$x == $y	Returns true if $x is equal to $y
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Increment / Decrement Operators		
Operator	**Name**	**Description**
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators			
Operator	**Name**	**Example**	**Result**
And (&&)	And	$x and $y or $x && $y	True if both $x and $y are true
Or (||)	Or	$x or $y or $x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators			
Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

- The if statement executes some code if one condition is true.

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

if ($t < "20") {
  echo "Have a good day!";
}
?>
```

- The if...else statement executes some code if a condition is true and another code if that condition is false.

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

if ($t < "20") {
  echo "Have a good day!";
}
else {
  echo "Have a good night!";
}
?>
```

- The if...elseif...else statement executes different codes for more than two conditions.

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

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

- Use the switch statement to select one of many blocks of code to be executed.

```php
<?php
$favcolor = "red";

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

- The while loop - Loops through a block of code as long as the specified condition is true.

```php
<?php
$x = 1;

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

- The do...while loop - Loops through a block of code once, and then repeats the loop as long as the specified condition is true.

```php
<?php
$x = 1;

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

- The for loop - Loops through a block of code a specified number of times.

```php
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

- An array can hold many values under a single name, and you can access the values by referring to an index number.
- In PHP, the array() function is used to create an array.

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

- In PHP, there are three types of arrays:
  - Indexed arrays - Arrays with a **numeric** index.
  - Associative arrays - Arrays with **named** keys.
- **PHP Indexed Arrays:**
  - The index can be assigned automatically (index always starts at 0), like this:

```php
$cars = array("Volvo", "BMW", "Toyota");
```

  - or the index can be assigned manually:

```php
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

- Loop through an Indexed Array:

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
  echo $cars[$x];
  echo "<br>";
}
?>
```

- **PHP Associative Arrays:**
  - Associative arrays are arrays that use named keys that you assign to them.

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

  - or:

```php
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

- **Printing arrays:**
  - The built-in function **`print_r()`** is used to print the value stored in a variable in PHP. We can also use it to print an array. It prints all the values of the array along with their index number.
  - **Echo** only prints the values of the array.
- **PHP Global Variables – Superglobals:**
  - Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class, or file without having to do anything special.
  - **PHP $_REQUEST** is a PHP super global variable which is used to collect data after submitting an HTML form (either post or get).

- Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class, or file without having to do anything special.
- **PHP $_POST** is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". **$_POST** is also widely used to pass variables.
- **PHP $_GET** is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get". **$_GET** can also collect data sent in the URL.

```
<html>
<body>

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

</body>
</html>
```

- To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

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

</body>
</html>
```

- The same result could also be achieved using the **HTTP GET** method.
- Both GET and POST create an array (e.g., array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
- Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL).
- GET should NEVER be used for sending passwords or other sensitive information!
- Information sent from a form with the POST method is **invisible** to others (all names/values are embedded within the body of the HTTP request)
- The **`isset()`** function is an inbuilt function in PHP which checks whether a variable is set and is not NULL.
- **Client side is not enough for validation, client and server side are important.**

- **PHP MySQL Database:**
  - The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows. Every table has a name
  - We can query a database for specific information and have a record set returned.
  - MySQL is a structured database. The database's name should have no spaces and it is case sensitive.
  - **MySQLi** extension (the "i" stands for improved)
  - Before we can access data in the MySQL database, we need to be able to connect to the server.

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_errno());
}
echo "Connected successfully";
?>
```

  - After connecting to the database, you should write the needed query (create, insert, update, or delete) and show any result if any.
  - The connection will be closed automatically when the script ends. To close the connection before, use the following:

```php
mysqli_close($conn);
```

  - Select Data from MySQL:

```php
$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE
lastname='Doe'";
```

  - To broaden the selections of a query, the percent sign (%) can be used.
  - Insert Data into MySQL:

```php
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

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

  - Delete Data from MySQL:

```php
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . mysqli_error($conn);
}
```

- Update Data in MySQL:

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . mysqli_error($conn);
}
```

- Full lecture MySQL example:

```
<table border=2 width="100%">
<tr><th>serial no</th><th>title</th><th>des</th></tr>
<?php
//connect to db
$counter=1;
$con=mysqli_connect("localhost","root","12345678","news_db");

//check the connection
if ($con)
{
    //write query
    $q=mysqli_query($con,"SELECT * FROM 'news_data'");
    if($q)
    {
        //fetch data into table
        while($row=mysqli_fetch_array($q)
        {
            echo "<tr>";
            echo "<td>".$counter++."</td>";
            echo "<td>".$row["news_title"]."</td>";
            echo "<td>".$row["news_details"]."</td>";
            echo "</tr>";
        }
    }
    else
    {
        echo "error select".mysqli_errno();
    }
die ("not connected".mysqli_errno());
}
//close connection
mysqli_close($con);
?>
</table>
```

- GET is easier that POST.
- If there are 500 HTML pages that have date, 500 page (that contain forms) should be made to accept that data and to pass [using GET or POST] to other pages.
- If there are 500 HTML pages only one database is needed to store data.
- mysqli affected_rows(): return the number of affected rows from different queries.

- There are 3 ways to navigate between pages:
  - HTML <a> href Attribute:
    - `<a href="https://www.w3schools.com">Visit W3Schools</a>`
  - PHP header:
    - `header("Location: select-all.php");`
- The `<input type="hidden">` defines a hidden input field.
  - A hidden field let web developers include data that cannot be seen or modified by users when a form is submitted.
  - A hidden field often stores what database record that needs to be updated when the form is submitted.
- **Php Sessions:**
  - A session is a way to store information (in variables) to be used across multiple pages [server side]. Unlike a cookie, the information is not stored on the user's computer [client side].
  - Session variables hold information about one single user and are available to all pages in one application.
  - A session is started with the `session_start()` function. It must be the very first thing in your document. Before any HTML tags.
  - Session variables are set with the PHP global variable: $_SESSION.

```php
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

  - Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (`session_start()`).

```php
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
```

  - To remove all global session variables and destroy the session, use `session_unset()` and `session_destroy()`.

- **PHP OOP:**
  - Every table should be in separate class. Table name should be the class name.
  - Check connection class in lecture codes on **Blackboard**.
  - A class is defined by using the `class` keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
?>
```

  - Objects of a class is created using the `new` keyword.

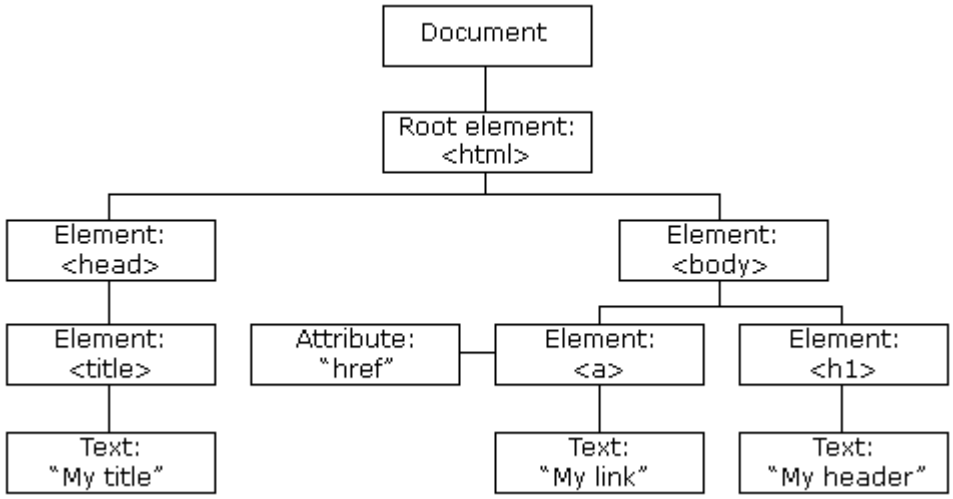
```php
$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');

echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
```

  - The $this keyword refers to the current object, and is only available inside methods.
  - An inherited class is defined by using the `extends` keyword.
  - require_once() function can be used to include a PHP file in another one.
  - parent::functionName(); → to use a function of the inherited class from another class.
- **Php security:**
  - Filtering is a process by which you inspect data to prove its validity.
  - `ctype_alnum()`: Checks if all of the characters in the provided string, text, are alphanumeric.
  - Escaping output is a process by which you escape characters that have a special meaning on a remote system.
  - `htmlentities()` — Convert all applicable characters to HTML entities.

- Register globals:
  - All super global variable array indexes are available as variable names.
  - A malicious user can include any script in your code.
  - Solution: Be aware that with register globals on, any user can inject a variable of any name into your PHP scripts.
- Spoofed forms:
  - Be aware that anybody can write their own forms and submit them to your PHP scripts.
  - Solution: Make sure all your rules are checked by the PHP external data filter, don't rely on a form to exert rules for you.
- Session fixation:
  - The malicious user is trying to 'steal' someone else's session on your site.
  - Solution: Regenerate the session id using this function `session_regenerate_id()` before any change in privilege level.
- SQL injection:
  - The goal of SQL injection is to insert arbitrary data, most often a database query, into a string that s eventually executed by the database.
  - Solution: Escape your data for a MySQL db, use the function `mysql_real_escape_string()`
- Accessing credentials:
  - Don't store passwords in an included file without a *.php extension.
  - You can store in a *.php file under the root.
  - Keep as much code as possible, including definition of passwords, in included files outside of the web accessible directories.
  - Use environment variables accessed in PHP.
- Cross-site scripting (XSS):
  - Malicious scripts are injected into trusted websites.
  - Be especially careful if you are writing user input to a file, which is later included into your page. Without checking, the user can then write their own PHP scripts for inclusion.
- Magic quotes:
  - If turned on, this automatically escapes quotation marks and backslashes in any incoming data. Although useful for beginners, it cannot be relied upon if you want to write portable code.
  - `get_magic_quotes_gpc()` function to tell if they are on or off.
  - To start from a consistent point, use `stripslashes()` to remove any escape characters added by 'magic quotes'.

JavaScript:

- JavaScript is a client-side programming.
- With the object model, JavaScript gets all the power it needs to create dynamic HTML.
- The **HTML DOM** model [**D**ocument **O**bject **M**odel] is constructed as a tree of **Objects**.
- If you want to access any element in an HTML page, you always start with accessing the **document** object, the document object represents your web page.



- In HTML, JavaScript code is inserted between `<script>` and `</script>` tags.
- Scripts can be placed in the `<body>`, or in the `<head>` section of an HTML page, or in both.
- HTML DOM **methods** are **actions** you can perform (on HTML Elements).
- HTML DOM **properties** are **values** (of HTML Elements) that you can set or change.
- JavaScript accepts both double and single quotes.
- Changing HTML elements:
  - `element.attribute = new value` → Change the attribute value of an HTML element.
  - `element.style.property = new style` → Change the style of an HTML element.
- Adding events handlers:
  - `document.getElementById(id).onclick = function(){code}` → Adding event handler code to an onclick event.
- One of many JavaScript HTML methods is `getElementById()`.

```
const element = document.getElementById("intro");
```

- Onmouseover: execute a JavaScript when moving the mouse pointer onto an image.
- Onmouseout: execute a JavaScript when moving the mouse pointer out of an image:

- This example changes the value of the src attribute of an `<img>` element:

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

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

- The following example changes the style of a `<p>` element:

```
<html>
<body>

<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>

</body>
</html>
```

- This example changes the style of the HTML element with id="id1", when the user clicks a button:

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

<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

</body>
</html>
```

- To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute: `onclick=`*`JavaScript`*
- In this example, the content of the <h1> element is changed when a user clicks on it:

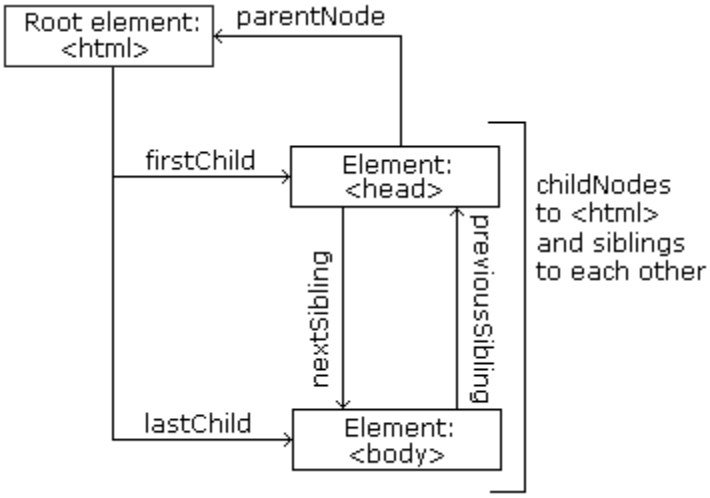
```
<!DOCTYPE html>
<html>
<body>
<h1 onclick="this.innerHTML = 'Ooops!'">Click on this text!</h1>
</body>
</html>
```

- In this example, a function is called from the event handler:

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

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
  id.innerHTML = "Ooops!";
}
</script>

</body>
</html>
```

JQuery:

- The terms parent, child, and sibling are used to describe the relationships.
  - In a node tree, the top node is called the root (or root node).
  - Every node has exactly one parent, except the root (which has no parent).
  - A node can have a number of children.
  - Siblings (brothers or sisters) are nodes with the same parent.



- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).
  - Basic syntax is: **$(selector).action()**
  - A $ sign to define/access jQuery
  - A (selector) to "query (or find)" HTML elements
  - A jQuery action() to be performed on the element(s)
    - `$(this).hide()` - hides the current element.
    - `$("p").hide()` - hides all <p> elements.
    - `$(".test").hide()` - hides all elements with class="test".

        - `$("#test").hide()` - hides the element with id="test".
- The Document Ready Event:
    - This is to prevent any jQuery code from running before the document is finished loading (is ready).

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

  // jQuery methods go here...

});
```

- When a user clicks on a button, all `<p>` elements will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

- When a user clicks on a button, the element with id="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

- When a user clicks on a button, the elements with class="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide(500);
  });
});
```

- The `prepend()` method inserts specified content at the **beginning** of the selected elements. Specifies the content to insert (can contain HTML tags).
- The `append()` method inserts specified content at the **end** of the selected elements.
- The `on()` method attaches one or more event handlers for the selected elements.

```
$("p").on("click", function(){
  $(this).hide();
});
```

- The jQuery `animate()` method is used to create custom animations.

```
$(selector).animate({params},speed,callback);
```

    - The required params parameter defines the CSS properties to be animated.
    - The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

```
$("button").click(function(){
  $("div").animate({left: '250px'},300);
});
```

HTML:

- Basic HTML Structure:

<html> [Creates an HTML document]

<head> [Sets off the title & other info that isn't displayed]

<title>website title</title> [Puts name of the document in the title bar]

</head>

<body> [Sets off the visible portion of the document]

content of webpage

</body>

</html>

Text tags:	
Tag:	**Meaning:**
<h?> .. </h?>	Heading (?= 1 for largest to 6 for smallest, eg h1)
paragraph	Paragraph of Text.
.. **..**	Make text between tags **bold.**
.. ..	Make text between tags *italic.*
 	Line Break (force a new line).
..	Text in center.
 	Sets size of font - 1 to 7.
 	Sets font color.

Graphical elements:	
Tag:	**Meaning:**
<img src="URL" height=? width=?>	Sets height and width of image.
clickable text	Creates a text hyperlink to a Uniform Resource Locator.
 <img src="URL" height=? width=?> 	Creates an image hyperlink to a Uniform Resource Locator.

Lists:	
Tag:	**Meaning:**
	Creates an unordered list.
<ul type="?">	Unordered list bullet type: disc, circle, square.
...	List Item (within ordered or unordered).
<ol type="?">	Ordered list type: A, a, I, i, 1.

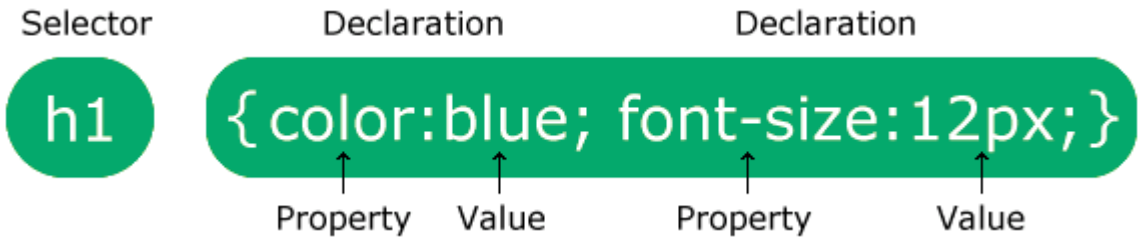
Tables:	
Tag:	**Meaning:**
<table> </table>	Creates a table.
<tr> </tr>	Sets off each row in a table.
<td> </td>	Sets off each cell in a row.
<th> </th>	Sets off the table header (a normal cell with bold, centered text)
<table border=?>	Sets the width of the border around table cells.
<table cellspacing=”?”>	Space between cells.
<table cellpadding=”?”>	Space between cell wall and content.
<td rowspan=?> </td>	Sets number of rows a cell should span.
<td colspan=?> </td>	Sets number of columns a cell should span.
<td bgcolor=”?” </td>	Set color of a cell.
<td> <img src="URL" height=? width=?> </td>	Set background of a cell.
<td align=?>	Sets alignment for cells (left/center/right).
<td valign=?>	Sets vertical alignment for cell (top/middle/bottom).

Forms:	
Tag:	**Meaning:**
<form> .. </form>	Form input group declaration.
<form action=”URL”>	URL of form script.
<form method=”***”>	Method of form: get, post.
<input type=”***”>	Input field type: text, radio, checkbox, submit, button and password.
<input value=”***”>	Value of input field
<input size=”***”>	Field size
<input maxlength=”***”>	Maximum length of input field data
<input hint=”***”>	Grey text that has no value.
<textarea name=? cols="x" rows="y"> </textarea>	Creates a text box area. Columns set the width; rows set the height.
<select> .. </select>	Used to create a drop-down list.
<option value=”..”> .. </option>	Used to display options for a drop-down list.

Others:	
<!-- .. -->	To make comment

CSS cheat sheet:

- A CSS rule consists of a selector and a declaration block.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.



- An internal CSS is defined in the `<head>` section of an HTML page, within a `<style>` element.
- The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powderblue" background color:

```
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.
- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the class name.
- The `:hover` selector is used to select elements when you mouse over them.
- In HTML, div and span tags are elements used to define parts of a document, so that they are identifiable when a unique classification is necessary.