الجامعة التكنولوجية
كلية علوم الحاسوب

Web Programming

Third Class
الفصل الثاني
2025-2026

MSc. Mohammed Thamer
Dr. Muna Ghazi
MSc. Fadhil Abbas

cs.uotechnology.e

Table of Contents

Week 1: Introduction to Web Development

1.1 What is a Web-Based Application?

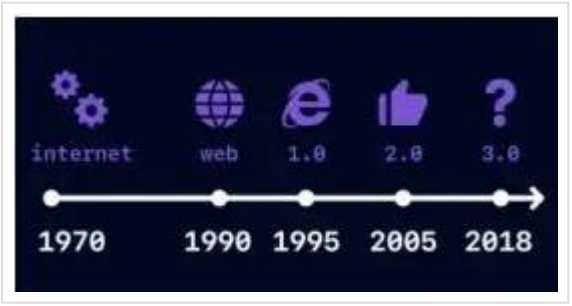
A web-based application, or web app, is a program that is stored on a remote server and delivered over the Internet through a browser interface. Unlike traditional desktop applications that are installed on a user's computer, web apps are accessed via a web browser like Chrome, Firefox, or Safari. This client-server model is fundamental to how the modern web works.

Examples of web applications are everywhere: online banking portals, social media sites like Facebook, webmail services like Gmail, and e-commerce stores like Amazon. They combine server-side scripting (e.g., PHP, Python) for logic and data processing with client-side scripting (e.g., JavaScript) for a dynamic and interactive user interface.

1.2 The Internet and the World Wide Web (WWW)

It's common to use "Internet" and "Web" interchangeably, but they are distinct concepts.

- **The Internet:** Often called the "network of networks,"; the Internet is the vast, global infrastructure of interconnected computer networks. It allows devices worldwide to communicate with each other. It supports many services, including email (SMTP), file transfer (FTP), and the World Wide Web.
- **The World Wide Web (WWW):** The Web is one of the services that runs on the Internet. It's a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia, and navigate between them via hyperlinks.



1.3 History and Growth of the Web

The Web was invented by Tim Berners-Lee at CERN in 1989. Initially conceived to meet the demand for automated information-sharing between scientists, it has grown exponentially. The early 90s saw the rise of static websites, which were simple HTML pages displaying fixed content. The introduction of technologies like CGI, and later server-side languages like PHP and client-side scripting with JavaScript, paved the way for dynamic, interactive web applications that dominate the internet today.

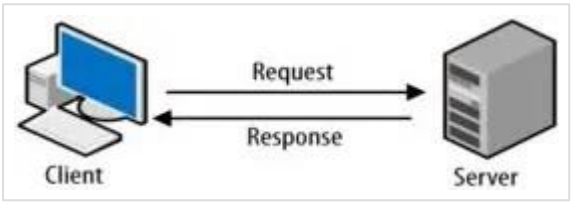
1.4 Core Concepts of Web Technology

- **Internet Service Provider (ISP):** An ISP is a company that provides individuals and organizations with access to the Internet and other related services. When you connect to the internet from home, you are using an ISP.
- **TCP/IP (Transmission Control Protocol/Internet Protocol):** This is the foundational communication protocol suite of the Internet. TCP/IP specifies how data should be packetized, addressed, transmitted, routed, and received. It ensures reliable, ordered, and error-checked delivery of a stream of data between applications.
- **HTTP (HyperText Transfer Protocol):** This is the protocol used for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers. When you type a URL in your browser, it sends an HTTP request to the server, which then returns an HTTP response containing the webpage content.
- **Web Browser:** A software application for accessing information on the World Wide Web. It interprets HTML, CSS, and JavaScript to render web pages for the user. Examples include Chrome, Firefox, Safari, and Edge.
- **Web Server:** A computer system that hosts websites. It runs web server software, such as Apache or Nginx, which processes incoming network requests over HTTP and serves the requested web page content to the user's browser.

1.5 The Client-Server Model

The client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. In the context of the web:

- **Client:** The user's web browser (e.g., Chrome, Firefox) that requests a web page.
- **Server:** The web server (e.g., Apache, Nginx) that stores the website's files and sends them to the client upon request.



1.6 Web Pages, Websites, and URLs

- **Web Page:** A single document, typically written in HTML, that is viewable on the Internet through a web browser.
- **Website:** A collection of related web pages, including multimedia content, typically identified with a common domain name, and published on at least one web server.
- **URL (Uniform Resource Locator):** The address of a resource on the Internet. It specifies the protocol to use, the server's domain name, and the specific file or resource path.

> ### Example: URL Structure
>
> `https://www.example.com/products/item.html`
>
> - **Protocol:** `https`
> - **Domain Name:** `www.example.com`
> - **Path:** `/products/item.html`

1.7 Website Classification

Websites can be classified in several ways:

- **By Functionality:**
  - **Static Websites:** Content is fixed and delivered to the user exactly as stored. Built with HTML and CSS.
  - **Dynamic Websites:** Content is generated on-the-fly by a server-side scripting language (like PHP) and often interacts with a database. This allows for personalized content, user accounts, and interactive features.
- **By Purpose:**
  - **E-commerce:** For selling goods and services (e.g., Amazon).
  - **Portfolio:** To showcase work (e.g., a photographer's site).
  - **Blog:** A regularly updated journal or informational website.
  - **Social Media:** For user interaction and content sharing (e.g., Facebook).
  - **Informational:** To provide information (e.g., Wikipedia).

Week 1 Homework

1. Define the Internet and the World Wide Web in your own words. What is the key difference?
2. Explain the roles of a client and a server in the context of browsing a website.
3. Break down the following URL into its components: `http://www.uotechnology.edu.iq/index.html`
4. Research and list three different Internet Service Providers (ISPs) available in your region.
5. Find three examples of dynamic websites and three examples of static websites. Explain why you classified them as such.

Week 2: HTML Fundamentals

2.1 Introduction to HTML

HTML stands for **HyperText Markup Language**. It is the standard markup language for creating web pages and web applications. Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. These elements are represented by **tags**. HTML tags label pieces of content such as "heading," "paragraph," "table," and so on. Browsers do not display the HTML tags, but use them to render the content of the page.

2.2 Basic HTML Document Structure

Every HTML document has a fundamental structure that includes the `<;!DOCTYPE html>` declaration, `<html>`, `<head>`, and `<body>` tags.

Basic HTML Page Structure

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
</head>
<body>

    <h1>My First Heading</h1>
    <p>My first paragraph.</p>

</body>
</html>
```

- The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document.
- The `<html>` element is the root element of an HTML page.
- The `<head>` element contains meta information about the document, such as its title.
- The `<title>` element specifies a title for the document, which is shown in the browser's title bar or in the page's tab.
- The `<body>` element contains the visible page content.

2.3 Common HTML Tags

Headings and Paragraphs

HTML headings are defined with the `<h1>` to `<h6>` tags. `<h1>` defines the most important heading. `<h6>` defines the least important heading. Paragraphs are defined with the `<p>` tag.

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Links (Anchors)

HTML links are defined with the `<a>` tag. The link's destination is specified in the `href` attribute.

```
<a href="https://www.example.com">Visit Example.com</a>
```

Images

HTML images are defined with the `<img>` tag. The `src` attribute specifies the path to the image, and the `alt` attribute provides alternate text.

```
<img src="image.jpg" alt="An example image" width="104"
height="142">
```

Lists

HTML supports ordered (numbered) and unordered (bulleted) lists.

List Examples

```
<h4>Unordered List</h4>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

<h4>Ordered List</h4>
<ol>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

2.4 Tables

HTML tables allow web developers to arrange data into rows and columns. A table is defined with the `<table>` tag. A table is divided into rows with the `<tr>` tag. Each row is divided into data cells with the `<td>` tag (table data). `<th>` is used for table headers.

Simple Table

```
<table border="1">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
```

```
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

2.5 Forms

HTML forms are used to collect user input. The `<form>` element is a container for different types of input elements, such as text fields, checkboxes, radio buttons, submit buttons, etc.

Simple Form

```
<form action="/action_page.php" method="post">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
</form>
```

2.6 Image Maps

An image map is an image with clickable areas. The `<map>` tag defines an image map. A map is an image with clickable areas. The areas are defined with one or more `<area>` tags.

Image Map Example

```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap">

<map name="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer"
```

```
href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone"
href="phone.htm">
  <area shape="circle" coords="337,300,44" alt="Coffee"
href="coffee.htm">
</map>
```

Week 2 Homework

1. Create an HTML page about your favorite hobby. It should include:
    - A main heading (`<h1>`) and at least two subheadings (`<h2>`).
    - Several paragraphs of text describing the hobby.
    - An image related to your hobby. Make sure to use the `alt` attribute.
    - An unordered list of things you need for this hobby.
    - An ordered list of steps to get started with this hobby.
2. Create a separate HTML page with a simple contact form. The form should have fields for Name, Email, and a Message (use a `<textarea>`). Include a "Submit" button.
3. On your hobby page, add a link to the contact form page.
4. Create a simple table on your hobby page that lists 3-4 items related to your hobby and their estimated cost. The table should have a header row.

Week 3: Advanced HTML & Introduction to CSS

3.1 Introduction to CSS

CSS stands for **Cascading Style Sheets**. It is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. It allows you to control the color, font, layout, and overall look of your web pages.

CSS Syntax

A CSS rule consists of a selector and a declaration block. The selector points to the HTML element you want to style. The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon.

3.2 Methods of Applying CSS

There are three ways to insert CSS into an HTML document:

1. **Inline CSS:** Uses the `style` attribute directly inside an HTML tag. This method is generally discouraged for large-scale styling as it mixes content with presentation.
2. **Internal CSS:** Defined within the `<head>` section of an HTML page, inside a `<style>` tag. This is useful for single pages that have a unique style.

3. **External CSS:** Styles are defined in a separate `.css` file. This is the most common and recommended method, as it allows you to change the look of an entire website by editing just one file. You link to it using the `<link>` tag in the `<head>` section.

CSS Application Methods

Inline CSS:

```
<h1 style="color:blue; text-align:center;">A Blue, Centered Heading</h1>
```

Internal CSS:

```
<head>
  <style>
    body {
      background-color: lightblue;
    }
    h1 {
      color: navy;
      margin-left: 20px;
    }
  </style>
</head>
```

External CSS:

In `styles.css`:

```
body {
  background-color: lightblue;
}
h1 {
  color: navy;
  margin-left: 20px;
}
```

In your HTML file:

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

```
</head>
```

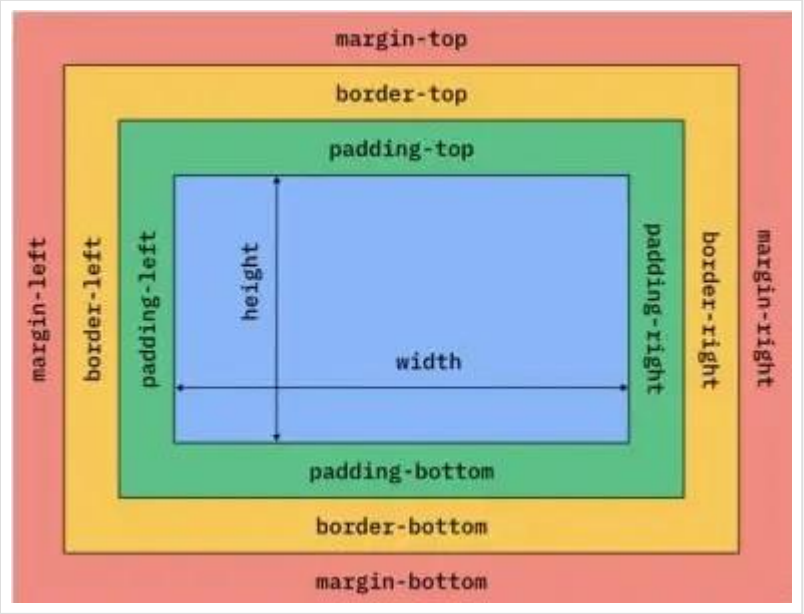
3.3 CSS Selectors

Selectors are used to "find" (or select) the HTML elements you want to style.

- **Element Selector:** Selects HTML elements based on the element name. (e.g., `p`, `h1`, `div`)
- **ID Selector:** Uses the `id` attribute of an HTML element to select a specific element. An ID should be unique within a page. (e.g., `#firstname`)
- **Class Selector:** Selects HTML elements with a specific class attribute. You can apply a class to multiple elements. (e.g., `.intro`)
- **Universal Selector:** Selects all HTML elements on the page. (`*`)
- **Grouping Selector:** Selects all the HTML elements with the same style definitions. (e.g., `h1, h2, p`)

3.4 The Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout. It is a box that wraps around every HTML element and consists of: margins, borders, padding, and the actual content.



- **Content:** The content of the box, where text and images appear.
- **Padding:** Clears an area around the content. The padding is transparent.
- **Border:** A border that goes around the padding and content.
- **Margin:** Clears an area outside the border. The margin is transparent.

3.5 Frameset (Legacy)

The `<frameset>` tag was used in older HTML versions to divide a web page into multiple sections or "frames". Each frame could load a separate HTML document. However, **framesets are now obsolete in HTML5** and should not be used. They have significant usability, accessibility, and SEO problems. Modern layouts are achieved with CSS.

> **Important:** While understanding what framesets were is useful for historical context, you should avoid using them in modern web development. Use CSS Flexbox, CSS Grid, or iframes (`<iframe>`) for embedding other documents.

```
<!-- This is an example of an obsolete frameset -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
  <head>
    <title>Frameset Example</title>
  </head>
  <frameset cols="25%,*,25%">
    <frame src="frame_a.htm">
    <frame src="frame_b.htm">
    <frame src="frame_c.htm">
  </frameset>
</html>
```

Week 3 Homework

1. Create an external CSS file named `style.css`.
2. Link this `style.css` file to the HTML page you created in Week 2.
3. In `style.css`, add rules to:
    - Set a background color for the entire page (`body`).
    - Set a different font family for all text on the page.
    - Center the main `<h1>` heading and give it a unique color.
    - Add a border, padding, and margin to the image on your page.
    - Style the links (`<a>` tags) to have a different color and remove the underline. Add a hover effect that changes the color and brings back the underline.

4. Create a new HTML page. In this page, use internal CSS (inside a `<style>` tag) to create three `<div>` elements. Style them to look like colored boxes (give them a specific height, width, and background color) and arrange them side-by-side. (Hint: You might need to research the `display` or `float` CSS properties).

Week 4: JavaScript Introduction

4.1 What is JavaScript?

JavaScript is a high-level, often just-in-time compiled, and multi-paradigm programming language. It is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2023, 98.7% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript enables interactive web pages and is an essential part of web applications. It can:

- Change HTML content and attributes.
- Change CSS styles.
- Validate user input in forms.
- React to user events (like clicks, mouse movements, key presses).
- Communicate with web servers asynchronously (AJAX).

4.2 Embedding JavaScript in HTML

JavaScript can be added to an HTML document in two main ways:

1. **Internal JavaScript:** By placing JavaScript code inside `<script>` and `</script>` tags. This can be in the `<head>` or `<body>` section. Placing scripts at the end of the `<body>` section is often recommended for better page load performance.
2. **External JavaScript:** By placing the code in a separate `.js` file and linking to it from the HTML document using the `src` attribute of the `<script>` tag. This is the preferred method for larger scripts as it promotes code reusability and organization.

Embedding JavaScript

Internal JavaScript:

```
<!DOCTYPE html>
<html>
<body>
    <h2>My First Web Page</h2>
    <p id="demo">A Paragraph.</p>
    <script>
        document.getElementById("demo").innerHTML = "Hello
JavaScript!";
    </script>
</body>
</html>
```

External JavaScript:

In `myScript.js`:

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph
changed.";
}
```

In your HTML file:

```
<!DOCTYPE html>
<html>
<body>
    <h2>My First Web Page</h2>
    <p id="demo">A Paragraph.</p>
    <button type="button" onclick="myFunction()">Try
it</button>
    <script src="myScript.js"></script>
</body>
</html>
```

4.3 JavaScript Variables and Data Types

Variables are containers for storing data values. In JavaScript, you can declare variables using `var`, `let`, or `const`.

- `var`: The historical way to declare variables. It has function scope.
- `let`: Introduced in ES6 (2015), allows you to declare variables with block scope.
- `const`: Also introduced in ES6, declares a block-scoped variable whose value cannot be changed.

JavaScript is a "loosely typed"; or "dynamically typed" language, meaning you don't have to declare the type of a variable. Common data types include: String, Number, Boolean, Object, and Undefined.

4.4 JavaScript Operators

Arithmetic Operators

Used to perform arithmetic on numbers:

- `+` (Addition), `-` (Subtraction), `*` (Multiplication), `/` (Division)
- `%` (Modulus - remainder of a division)
- `++` (Increment), `--` (Decrement)

> #### Arithmetic Example
>
> ```
> let x = 10;
> let y = 5;
> let z = x + y; // z will be 15
> ```

Logical Operators

Used to determine the logic between variables or values:

- `&&` (Logical AND)
- `||`(Logical OR)
- `!` (Logical NOT)

Logical Operator Example

```
let age = 25;
let hasLicense = true;

if (age > 18 && hasLicense) {
  console.log("Can legally drive.");
}
```

4.5 Conditional Statements

Conditional statements are used to perform different actions based on different conditions. The most common are `if`, `else if`, and `else`.

Conditional Statement Example

```
let time = new Date().getHours();
let greeting;

if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}

console.log(greeting);
```

4.6 Pop-up Boxes

JavaScript has three kinds of pop-up boxes: Alert, Confirm, and Prompt.

- **Alert Box:** Used to ensure information gets to the user.

```
alert("Hello! I am an alert box!");
```

- **Confirm Box:** Used to let the user verify or accept something. It returns `true` for OK and `false` for Cancel.

```
let userChoice = confirm("Press a button!");
if (userChoice == true) {
  // User pressed OK
} else {
  // User pressed Cancel
}
```

- **Prompt Box:** Used to prompt the user for input. It returns the input value if the user clicks OK, and `null` if the user clicks Cancel.

```
let person = prompt("Please enter your name", "Harry Potter");
if (person != null) {
  console.log("Hello " + person + "!");
}
```

Week 4 Homework

1. Create an HTML file with a button. When the button is clicked, use JavaScript to display an alert box that says "Hello, World!".
2. Write a JavaScript program that asks the user for their name using a `prompt()` box and then displays a welcome message in an alert box, like "Welcome, [Name]!".
3. Create a script that asks the user for two numbers using two separate `prompt()` boxes. Then, calculate and display the sum, difference, product, and quotient of these two numbers in the browser console (use `console.log()`).
4. Write a script that checks if a number entered by the user is positive, negative, or zero. Display the result in an alert box.

5. Create a "Confirm" dialog that asks "Are you sure you want to leave this page?". If the user clicks "OK", redirect them to "https://www.google.com". If they click "Cancel", display an alert saying "You chose to stay!".

Week 5: JavaScript Control Flow, Functions, and Arrays

5.1 Loops

Loops are handy if you want to run the same code over and over again, each time with a different value. JavaScript supports different kinds of loops:

- **for** - loops through a block of code a number of times.
- **for/in** - loops through the properties of an object.
- **for/of** - loops through the values of an iterable object.
- **while** - loops through a block of code while a specified condition is true.
- **do/while** - also loops through a block of code while a specified condition is true, but the block is executed at least once.

Loop Examples

For Loop:

```
for (let i = 0; i < 5; i++) {
  console.log("The number is " + i);
}
```

While Loop:

```
let i = 0;
while (i < 5) {
  console.log("The number is " + i);
  i++;
}
```

Do/While Loop:

```
let i = 0;
do {
  console.log("The number is " + i);
  i++;
} while (i < 5);
```

5.2 JavaScript Functions

A JavaScript function is a block of code designed to perform a particular task. A function is executed when "something" invokes it (calls it). Functions are fundamental for writing reusable and organized code.

Function Examples

Defining and Calling a Function:

```
function greet(name) {
  return "Hello, " + name + "!";
}

let message = greet("Alice"); // Call the function
console.log(message); // Outputs: Hello, Alice!
```

Function Expression:

```
const multiply = function(a, b) {
  return a * b;
};

let product = multiply(4, 5); // product is 20
```

5.3 Lifetime of JavaScript Variables (Scope)

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes: global and local.

- **Global Scope:** A variable declared outside a function becomes GLOBAL. All scripts and functions on a web page can access it.
- **Local Scope:** A variable declared within a JavaScript function becomes LOCAL to the function. It can only be accessed from within that function.
- **Block Scope (let and const):** Variables declared with `let` or `const` inside a block `{}` cannot be accessed from outside the block.

Scope Example

```
let carName = "Volvo"; // Global scope

function myFunction() {
  let carName = "BMW"; // Local scope
  console.log("Inside function: " + carName); // Outputs: BMW
}

myFunction();
console.log("Outside function: " + carName); // Outputs: Volvo
```

5.4 Event Handlers

HTML events are ";things" that happen to HTML elements. When JavaScript is used in HTML pages, JavaScript can "react" on these events. An HTML event can be something the browser does, or something a user does.

Common HTML events:

- `onclick` - The user clicks an HTML element
- `onmouseover` / `onmouseout` - The user moves the mouse over/out of an element
- `onkeydown` - The user presses a key
- `onload` - The browser has finished loading the page
- `onsubmit` - The form is submitted

Event Handler Example

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

```
<body>

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

<button onclick="displayDate()">The time is?</button>
<p id="demo"></p>

<script>
function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
</script>

</body>
</html>
```

5.5 Arrays

An array is a special variable, which can hold more than one value at a time. If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
let car1 = "Saab";
let car2 = "Volvo";
let car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300? The solution is an array!

Array Example

```
// Creating an array
let cars = ["Saab", "Volvo", "BMW"];

// Accessing elements
console.log(cars[0]); // Outputs: Saab

// Changing an element
cars[0] = "Opel";
console.log(cars[0]); // Outputs: Opel
```

```
// Array properties and methods
console.log(cars.length);     // The length of an array
cars.sort();                  // Sorts the array
cars.push("Toyota");          // Adds a new element to the end
cars.pop();                   // Removes the last element
```

Week 5 Homework

1. **Quiz on Loops:** Write a short quiz with 3 multiple-choice questions about loops. Use `prompt()` to ask the questions and `alert()` to tell the user if they got it right or wrong. Keep track of their score and display it at the end.
2. **Function Practice:** Write a function called `calculateArea` that takes two parameters, `width` and `height`, and returns the area of a rectangle. Call this function with different values and log the results to the console.
3. **Event Handling:** Create an HTML page with a single `
   ` element. Using JavaScript and event handlers, make the div change its background color to red when the mouse pointer moves over it, and back to its original color when the mouse pointer moves out.
4. **Array Manipulation:** Create an array of your 5 favorite foods.
   - Use a `for` loop to print each food item to the console.
   - Add a new favorite food to the end of the array.
   - Remove the first food item from the array.
   - Print the final, modified array to the console.

Week 6: Advanced JavaScript - Strings and Objects

6.1 String Methods

Strings are fundamental for storing and manipulating text. JavaScript provides a rich set of built-in methods to work with strings.

Common String Methods

```
let text = "Hello World, this is a test string.";

// Length of the string
console.log("Length:", text.length); // Output: 34

// Finding a substring
console.log("Index of 'World':", text.indexOf("World")); //
Output: 6
console.log("Last index of 'is':", text.lastIndexOf("is")); //
Output: 18

// Extracting parts of a string
console.log("Slice:", text.slice(6, 11)); // Output: World
console.log("Substring:", text.substring(6, 11)); // Output:
World
console.log("Substr:", text.substr(6, 5)); // Output: World

// Replacing content
let newText = text.replace("World", "Universe");
console.log("Replaced:", newText); // Output: Hello Universe,
this is a test string.

// Case conversion
console.log("Uppercase:", text.toUpperCase());
console.log("Lowercase:",  text.toLowerCase());
```

```
//  Concatenation
let text1 = "Hello";
let text2 = "World";
let text3 = text1.concat(" ", text2);
console.log("Concatenated:", text3); // Output: Hello World

// Trimming whitespace
let paddedText = "    Hello World!    ";
console.log("Trimmed:", paddedText.trim()); // Output: Hello
World!

// Accessing characters
console.log("Character at index 4:", text.charAt(4)); //
Output: o
console.log("Character code at index 4:", text.charCodeAt(4));
// Output: 111

// Splitting a string into an array
let words = text.split(" ");
console.log("Words array:", words); // Output: ["Hello",
"World,", "this", "is", "a", "test", "string."]
```

6.2 Inserting Special Characters

To use special characters like single quotes, double quotes, or backslashes within a string, you need to "escape" them using a backslash (`\`).

Escaping Characters

```
let text1 = "We are the so-called \"Vikings\" from the
north.";
console.log(text1); // Output: We are the so-called "Vikings"
from the north.

let text2 = 'It\'s alright.';
console.log(text2); // Output: It's alright.

let text3 = "The character \\ is called backslash.";
```

```
console.log(text3); // Output: The character \ is called
backslash.
```

6.3 Creating New Objects

JavaScript is an object-based language. Almost everything is an object. Objects are variables too, but they can contain many values. The values are written as **name:value** pairs (name and value separated by a colon).

Object Creation

```
// Using an object literal (most common)
const person = {
  firstName: "John",
  lastName: "Doe",
  age:  50,
  eyeColor: "blue",
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
};

console.log(person.firstName); // Accessing property
console.log(person.fullName()); // Calling a method

// Using the 'new' keyword
const car = new Object();
car.make = "Toyota";
car.model = "Camry";
car.year  =  2022;
car.start = function() {
  console.log("Engine started!");
};

car.start();
```

6.4 The Date Object

The `Date` object is a built-in object in JavaScript that works with dates and times. You can create a new Date object using `new Date()`.

Working with Dates

```
// Create a new Date object for the current date and time
const now = new Date();
console.log(now.toString());

// Create a Date object for a specific date
const specificDate = new Date("2024-12-25");
console.log(specificDate.toDateString());

// Getting components of a date
console.log("Full Year:", now.getFullYear()); // e.g., 2024
console.log("Month (0-11):", now.getMonth()); // e.g., 6 for
July
console.log("Date:", now.getDate()); // e.g., 15
console.log("Day of week (0-6):", now.getDay()); // e.g., 1
for Monday
console.log("Hours:", now.getHours());
console.log("Minutes:", now.getMinutes());
console.log("Seconds:", now.getSeconds());

// Setting components of a date
const futureDate = new Date();
futureDate.setFullYear(2025);
futureDate.setMonth(11); // December
futureDate.setDate(31);
console.log("Future Date:", futureDate.toDateString());
```

Week 6 Homework

1. **String Manipulator:** Create a function that takes a string as input. The function should:
    - Convert the string to all uppercase.
    - Replace all instances of the letter 'A' with the number '4'.
    - Return the modified string.
    - Test it with a sample sentence and log the result.

2. **Object Creator:** Create an object representing a book. It should have properties for `title`, `author`, `yearPublished`, and `pages`. It should also have a method called `getSummary()` that returns a string like: "The book 'Title' was written by Author in Year."
3. **Date Formatter:** Write a function that creates a new `Date` object for the current time and returns a formatted string like "Today is Monday, July 15, 2024. The current time is 14:30:05."
4. **Character Counter:** Write a function that takes two arguments: a string and a character. The function should count and return how many times the specified character appears in the string. For example, `countChar("hello world", "l")` should return 3.

Week 7: JavaScript Math Object and Regular Expressions

7.1 The Math Object

The JavaScript `Math` object allows you to perform mathematical tasks on numbers. It is not a constructor, so you don't use the `new` keyword. All properties and methods of Math are static and can be called by using `Math` as an object.

Math Properties (Constants)

```
console.log(Math.E);          // returns Euler's number
console.log(Math.PI);         //  returns  PI
console.log(Math.SQRT2);      // returns the square root of 2
console.log(Math.SQRT1_2);   // returns the square root of 1/2
console.log(Math.LN2);        // returns the natural logarithm of 2
console.log(Math.LN10);       // returns the natural logarithm of 10
console.log(Math.LOG2E);      // returns base 2 logarithm of E
console.log(Math.LOG10E);     // returns base 10 logarithm of E
```

Math Methods

The syntax for any Math method is: `Math.method(number)`

Common Math Methods

```
// Number to Integer
Math.round(4.7);     // returns 5
Math.ceil(4.4);      // returns 5 (rounds up)
Math.floor(4.7);     // returns 4 (rounds down)
Math.trunc(4.7);     // returns 4 (returns the integer part)
```

```
// Power and Root
Math.pow(8, 2);       // returns 64 (8 to the power of 2)
Math.sqrt(64);      // returns 8

// Absolute Value
Math.abs(-4.7);     // returns 4.7

// Min and Max
Math.min(0, 150, 30, 20, -8, -200);  // returns -200
Math.max(0, 150, 30, 20, -8, -200);  // returns 150

// Random Number
Math.random();      // returns a random number between 0
(inclusive), and 1 (exclusive)

// Example: Get a random integer between 1 and 10
let randomInt = Math.floor(Math.random() * 10) + 1;
console.log(randomInt);
```

7.2 Regular Expressions

A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern. They can be used to perform all types of text search and text replace operations.

Syntax

```
/pattern/modifiers;
```

Example: `/w3schools/i` is a regular expression. `w3schools` is a pattern (to be used in a search), and `i` is a modifier (modifies the search to be case-insensitive).

Using String Methods with Regular Expressions

In JavaScript, regular expressions are often used with the two string methods: `search()` and `replace()`.

- The `search()` method uses an expression to search for a match, and returns the position of the match.
- The `replace()` method returns a modified string where the pattern is replaced.

Using `search()` and `replace()`

```
let str = "Visit W3Schools!";
let n = str.search(/w3schools/i); // n will be 6

let res = str.replace(/w3schools/i, "Microsoft"); // res will
be "Visit Microsoft!"
```

Regular Expression Modifiers

- `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:

- `[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:

- `\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:

- `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

Using the `test()` Method

The `test()` method is a RegExp expression method. It searches a string for a pattern, and returns true or false, depending on the result.

```
const pattern = /e/;
pattern.test("The best things in life are free!"); // Returns
true

const pattern2 = /\\d/; // a digit
pattern2.test("I have 5 apples."); // Returns true
```

Week 7 Homework

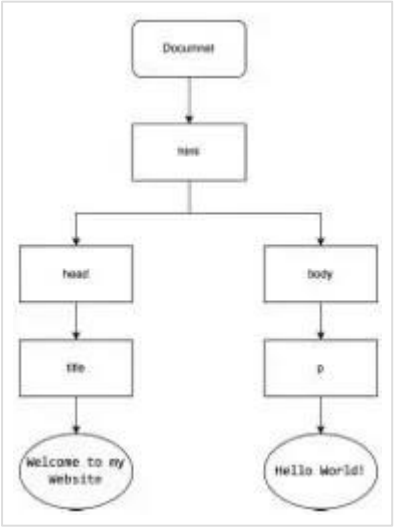
1. **Random Number Game:**
    - Generate a random integer between 1 and 100.
    - Create a simple HTML page with an input field and a "Guess" button.
    - Write JavaScript code that takes the user's guess from the input field when they click the button.
    - Compare the user's guess to the random number.
    - Display a message to the user: "Too high!", "Too low!", or "You got it!".
2. **Email Validator:**
    - Create an HTML page with a text input for an email address and a "Validate" button.
    - Write a JavaScript function that uses a regular expression to check if the entered text is a valid email format.
    - When the button is clicked, display a message indicating whether the email is valid or not.
3. **Password Strength Checker:**
    - Create an HTML page with a password input field.
    - As the user types in the password field (use the `onkeyup` event), use regular expressions to check for:
        - At least 8 characters long
        - Contains at least one number
        - Contains at least one uppercase letter
        - Contains at least one lowercase letter

- Display messages to the user indicating which criteria they have met.

Week 8: Form Validation and DOM Manipulation

8.1 The Document Object Model (DOM)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page. The HTML DOM model is constructed as a tree of Objects. It provides a structured representation of the document, allowing you to modify its content and visual presentation using a scripting language like JavaScript.



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page.
- JavaScript can change all the HTML attributes in the page.
- JavaScript can change all the CSS styles in the page.
- JavaScript can remove existing HTML elements and attributes.
- JavaScript can add new HTML elements and attributes.
- JavaScript can react to all existing HTML events in the page.
- JavaScript can create new HTML events in the page.

8.2 Finding HTML Elements

To manipulate an HTML element, you first need to find it. There are several ways to do this:

- `document.getElementById(id)`: Finds an element by its unique ID.
- `document.getElementsByTagName(name)`: Finds all elements with the specified tag name.
- `document.getElementsByClassName(name)`: Finds all elements with the specified class name.
- `document.querySelector(cssSelector)`: Finds the first element that matches a specified CSS selector.
- `document.querySelectorAll(cssSelector)`: Finds all elements that match a specified CSS selector.

Using `getElementById`

The `getElementById()` method is one of the most common ways to access an HTML element. It returns the element as an object.

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

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

<script>
  const element = document.getElementById("intro");
  // Now you can manipulate the element
  element.innerHTML = "New text!";
  element.style.color = "blue";
</script>

</body>
</html>
```

8.3 Changing HTML Content and Styles

Once you have an element object, you can change its content and style.

- `element.innerHTML`: Changes the inner HTML (content) of an element.

- `element.attribute`: Changes the attribute value of an HTML element (e.g., `element.src` for an image).
- `element.style.property`: Changes the style of an HTML element (e.g., `element.style.color`).

Changing Content and Style

```
<p id="p1">This is a paragraph.</p>

<script>
  const p1 = document.getElementById("p1");
  p1.innerHTML = "New text has been added!";
  p1.style.fontFamily = "Arial";
  p1.style.fontSize = "20px";
</script>
```

8.4 Form Validation

HTML form validation can be done by JavaScript. It provides a way to check user-entered data on the client-side before it is sent to the server. This provides immediate feedback to the user and reduces server load.

Simple Form Validation

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

</head>
<body>

<h2>JavaScript Validation</h2>

<form name="myForm" onsubmit="return validateForm()"
method="post">
  Name: <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>
```

```
<script>
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false; // Prevents the form from being submitted
  }
}
</script>

</body>
</html>
```

Data Validation Example

This example demonstrates how to validate numeric input to be between 1 and 10.

```
<p>Please input a number between 1 and 10:</p>
<input id="numb">
<button type="button" onclick="myFunction()">Submit</button>
<p id="demo"></p>

<script>
function myFunction() {
  // Get the value of the input field with id="numb"
  let x = document.getElementById("numb").value;
  // Get the element with id="demo"
  let text;
  if (isNaN(x) || x < 1 || x > 10) {
    text = "Input not valid";
  } else {
    text = "Input OK";
  }
  document.getElementById("demo").innerHTML = text;
}
</script>
```

Week 8 Homework

1. **Interactive Page:** Create an HTML page with a paragraph of text and three buttons: "Change Color", "Change Font Size", and "Hide".
    - Use `getElementById` to select the paragraph.
    - When "Change Color" is clicked, change the text color to a random color.
    - When "Change Font Size" is clicked, increase the font size by 2 pixels.
    - When "Hide" is clicked, make the paragraph disappear. Add a "Show" button that appears, which makes the paragraph visible again.
2. **Form Validation Project:** Create a user registration form with the following fields:
    - Username (must be at least 6 characters)
    - Email (must be a valid email format)
    - Password (must be at least 8 characters)
    - Confirm Password (must match the password)

    Write a JavaScript function that is called when the form is submitted. This function should:
    1. Prevent the form from submitting if any validation fails.
    2. Display a specific error message next to each field that has an error (e.g., "Username must be at least 6 characters long").
    3. If all fields are valid, display a success message like "Registration successful!".

Week 9: Midterm Exam

This week is dedicated to the midterm examination. The exam will cover all topics from Week 1 to Week 8, including fundamental web concepts, HTML, CSS, and JavaScript.

Topics to Review

- **Week 1:** Client-Server model, HTTP, URL structure, difference between Internet and Web.
- **Week 2:** Core HTML tags (`<h1>` , `<p>` , `<a>` , `<img>`), lists, tables, and forms.
- **Week 3:** CSS syntax, selectors (ID, class, element), the box model, and methods of applying CSS (inline, internal, external).
- **Week 4:** JavaScript basics, variables (`var` , `let` , `const`), data types, and operators (arithmetic, logical).
- **Week 5:** Control flow (if/else, switch), loops (for, while), functions, and event handling.
- **Week 6:** String methods, object creation, and the Date object.
- **Week 7:** The Math object and using regular expressions for pattern matching.

- **Week 8:** DOM manipulation (`getElementById`, `innerHTML`, `style`) and client-side form validation.

Sample Exam Questions

Short Answer:

Explain the difference between `==` and `===` in JavaScript. Provide an example.

Code Correction:

Find and fix the error in the following JavaScript code:

```
function showMessage() {
  document.getElementByID("message").innerHTML = "Hello";
}
```

Practical Problem:

Write the HTML and JavaScript for a button that, when clicked, changes the background color of the page to a random color.

Week 10: PHP Introduction and Basics

10.1 Introduction to PHP

PHP (Hypertext Preprocessor) is a widely-used open-source server-side scripting language designed for web development. PHP scripts are executed on the server, and the result is returned to the browser as plain HTML. This means that the user's browser receives standard HTML, and the PHP code itself is never visible to the client.

Why Learn PHP?

- **Free and Open Source:** No licensing fees are required.
- **Easy to Learn:** The syntax is similar to C, Java, and Perl, making it easy for programmers to pick up.
- **Cross-Platform:** Runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.).
- **Server Compatibility:** Compatible with almost all modern servers (Apache, Nginx, IIS, etc.).
- **Database Support:** Supports a wide range of databases, most notably MySQL/MariaDB.
- **Large Community:** A vast community means extensive documentation, tutorials, and support are readily available.

10.2 Setting Up a PHP Environment

To start developing with PHP, you need a local server environment. This typically includes:

- **A web server:** Apache is the most common choice.
- **PHP:** The PHP interpreter itself.
- **A database:** MySQL or MariaDB is the standard choice for PHP development.

The easiest way to set this up is by using a pre-packaged software stack like **XAMPP** (for Windows, macOS, and Linux) or **WAMP** (for Windows). These packages install and configure Apache, PHP, and MySQL for you.

10.3 PHP Syntax Basics

PHP code is executed on the server. It is embedded within HTML using special processing instructions, `<?php ... ?>`. The server processes the PHP code and sends the resulting HTML to the browser.

> ### Example 10.1: Basic PHP Syntax
>
> ```
> <!DOCTYPE html>
> <html>
> <body>
>     <h1>My First PHP Page</h1>
>     <?php
>         echo "Hello, World!";
>     ?>
> </body>
> </html>
> ```
>
> In the example above, `echo` is a language construct that outputs one or more strings.

Important Syntax Rules:

- PHP statements must end with a semicolon (`;`).
- Comments can be written as `// This is a single-line comment` or `# This is also a single-line comment`.
- Multi-line comments are enclosed in `/* ... */`.
- Variable names are case-sensitive (`$name` and `$Name` are different).
- Function names, class names, and keywords (e.g., `if`, `else`, `while`) are **not** case-sensitive.

10.4 Variables in PHP

Variables in PHP are used to store information. A variable starts with the `$` sign, followed by the name of the variable.

Example 10.2: Declaring and Using Variables

```
<?php
$greeting = "Hello";
$name = "Ahmed";
$age = 25;
$price = 99.99;

// The dot (.) is the concatenation operator
echo $greeting . ", " . $name . "!<br>";
echo "You are " . $age . " years old.<br>";

// You can also embed variables directly in double-quoted
strings
echo "The price is $price dollars.";
?>
```

Variable Scope

The scope of a variable is the context within which it is defined. PHP has several variable scopes:

- **local:** A variable declared inside a function has a local scope and can only be accessed within that function.
- **global:** A variable declared outside a function has a global scope and can only be accessed outside a function. To use it inside a function, you must use the `global` keyword.
- **static:** When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable NOT to be deleted. You can use the `static` keyword for this.

Example 10.3: Variable Scope

```
<?php
$x = 5; // global scope
```

```php
function myTest() {
  // using x inside this function will generate an error
  // echo "<p>Variable x inside function is: $x</p>";

  global $x; // use the global keyword
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

10.5 Data Types and Operators

PHP supports the following data types: String, Integer, Float (floating-point numbers - also called double), Boolean, Array, Object, NULL, and Resource.

PHP operators are similar to those in JavaScript and other C-style languages.

- **Arithmetic:** `+`, `-`, `*`, `/`, `%` (modulus), `**` (exponentiation).
- **Assignment:** `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `.=` (concatenation assignment).
- **Comparison:** `==` (equal), `===` (identical), `!=` (not equal), `<>` (not equal), `!==` (not identical), `>`, `<`, `>=`, `<=`, `<=>` (spaceship).
- **Logical:** `and`, `or`, `xor`, `&&`, `||`, `!`.

Example 10.4: Operators in Action

```php
<?php
$a = 10;
$b = "10";
$c = 5;

// Comparison
var_dump($a == $b);  // bool(true) - values are equal
var_dump($a === $b); // bool(false) - types are not equal

// Logical
$is_adult = true;
$has_permission = false;
if ($is_adult && $has_permission) {
    echo "Access granted.";
```

```
} else {
    echo "Access denied.";
}
echo "<br>";

// String Concatenation
$firstName = "John";
$lastName = "Doe";
$fullName = $firstName . " " . $lastName;
echo $fullName;
?>
```

Week 10 Homework

1. Set up a local development environment (like XAMPP) on your computer. Create a new file named `info.php` in your web server's root directory (e.g., `htdocs` for XAMPP) and add the following code: `<?php phpinfo(); ?>`. Access it in your browser (e.g., `http://localhost/info.php`) to verify your PHP installation.
2. Create a PHP script that declares variables for your name, city, and country. Display a sentence like "My name is [Name], and I live in [City], [Country]."
3. Define a constant for the value of PI (3.14159). Write a script that calculates and displays the area of a circle with a radius of 5 (Area = PI * r^2).
4. Write a script that takes two numbers, `$num1 = 15` and `$num2 = 4`. Display the results of their addition, subtraction, multiplication, division, and modulus.
5. Using string operators, create a single variable `$story` that combines the following strings into a coherent sentence: `"PHP is"`, `" a popular"`, `" scripting language."`. Then, use `echo` to display the final sentence.

Week 11: PHP Control Structures, Functions, and Arrays

11.1 Control Structures

Control structures are the core of any programming language, allowing you to control the flow of execution based on certain conditions or to repeat a block of code multiple times.

Conditional Statements: `if`, `elseif`, `else`

These statements allow your program to execute different code blocks based on whether a condition is true or false.

Example 11.1: Grading System

```php
<?php
$score = 85;
$grade = '';

if ($score >= 90) {
    $grade = 'A';
} elseif ($score >= 80) {
    $grade = 'B';
} elseif ($score >= 70) {
    $grade = 'C';
} elseif ($score >= 60) {
    $grade = 'D';
} else {
    $grade = 'F';
}
```

```
echo "Your score is $score, which is a grade of $grade.";
?>
```

Switch Statement

The `switch` statement is used to perform different actions based on different conditions/values. It is a good alternative to a long series of `if-elseif-else` statements.

Example 11.2: Day of the Week

```
<?php
$today = date("D"); // Gets the three-letter name of the day
(e.g., "Mon")

switch ($today) {
    case "Mon":
        echo "Today is Monday. Start of the work week!";
        break;
    case "Fri":
        echo "Today is Friday. The weekend is near!";
        break;
    case "Sat":
    case "Sun":
        echo "Have a great weekend!";
        break;
    default:
        echo "Have a nice day!";
}
?>
```

Loops: `while`, `do-while`, `for`, `foreach`

Loops are used to execute the same block of code a specified number of times or while a specified condition is true.

Example 11.3: Loop Examples

```php
<?php
// while loop
echo "While loop: ";
$i = 1;
while ($i <= 5) {
    echo $i . " ";
    $i++;
}
echo "<br>";

// do-while loop
echo "Do-while loop: ";
$i = 1;
do {
    echo $i . " ";
    $i++;
} while ($i <= 5);
echo "<br>";

// for loop
echo "For loop: ";
for ($j = 1; $j <= 5; $j++) {
    echo $j . " ";
}
echo "<br>";

// foreach loop (for arrays)
echo "Foreach loop: ";
$colors = array("red", "green", "blue");
foreach ($colors as $value) {
    echo $value . " ";
}
echo "<br>";
?>
```

11.2 Functions

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

Example 11.4: User-Defined Function

```
<?php
// Function with parameters and a return value
function calculateSum($num1, $num2) {
    $total = $num1 + $num2;
    return $total;
}

$result = calculateSum(10, 25); // Call the function
echo "The sum is: " . $result; // Outputs: The sum is: 35

// Function with a default argument
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350); // Outputs: The height is : 350
setHeight();     // Outputs: The height is : 50
?>
```

11.3 Arrays

An array stores multiple values in one single variable. PHP has three types of arrays:

- **Indexed arrays:** Arrays with a numeric index.
- **Associative arrays:** Arrays with named keys.
- **Multidimensional arrays:** Arrays containing one or more other arrays.

Example 11.5: Array Types

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

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

```php
// Multidimensional array
$students = array(
    array("John", 22, "A"),
    array("Jane", 20, "B"),
    array("Doe", 25, "C")
);

echo $students[0][0] . ": Age " . $students[0][1] . ", Grade "
. $students[0][2] . ".<br>";

// Looping through an associative array
foreach($age as $key => $value) {
    echo "Key=" . $key . ", Value=" . $value . "<br>";
}
?>
```

Week 11 Homework

1. Write a PHP script that uses a `for` loop to print the multiplication table for the number 7 (from 7x1 to 7x10).
2. Create a function named `isLeapYear` that takes a year as a parameter and returns `true` if it's a leap year and `false` otherwise. A leap year is divisible by 4, but not by 100 unless it is also divisible by 400. Test it with the years 2000, 2023, and 2024.
3. Create an associative array to store the details of a student (e.g., `name`, `id`, `major`, `gpa`). Use a `foreach` loop to print out the student's details in a user-friendly format (e.g., "Name: John Doe").
4. Write a function that takes an array of numbers as input and returns the average of those numbers.

Week 12: Forms, File Handling, and Validation

12.1 Handling HTML Forms with PHP

A key feature of PHP is its ability to collect data from HTML forms. The data is sent to the server, where a PHP script can process it. The two main methods for sending form data are GET and POST.

- **GET:** Form data is sent as URL variables. It's visible to everyone in the URL. There is a limit to the amount of information you can send.
- **POST:** Form data is sent inside the body of the HTTP request. It's not visible in the URL, and there are no limits on the amount of information to send.

PHP provides two superglobal variables, `$_GET` and `$_POST`, to access the form data.

Example 12.1: A Simple Form

HTML (index.php):

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

PHP (welcome.php):

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

12.2 Form Validation and Sanitization

It is crucial to validate and sanitize all user input to prevent security vulnerabilities like Cross-Site Scripting (XSS) and to ensure data integrity.

- **Validation:** Ensures the data is in the correct format (e.g., a valid email address).
- **Sanitization:** Removes any illegal characters or scripts from the data.

Example 12.2: Validation and Sanitization

```
<?php
// define variables and set to empty values
$name = $email = $comment = "";
$nameErr = $emailErr = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
```

```
    }
  }

  $comment = test_input($_POST["comment"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<!-- The HTML form would then display the error messages -->
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40">
</textarea>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>
```

12.3 File Handling

PHP has several functions for creating, reading, uploading, and editing files.

Reading a File

The `readfile()` function reads a file and writes it to the output buffer. The `fopen()`, `fread()`, and `fclose()` functions give you more control over file handling.

> **Example 12.3: Reading a File**

```php
<?php
// Method 1: Read entire file into a string
$content = file_get_contents("webdictionary.txt");
echo $content;

// Method 2: Read file line by line
$myfile = fopen("webdictionary.txt", "r") or die("Unable to
open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

Writing and Appending to a File

The `fwrite()` function is used to write to a file. The first parameter of `fwrite()` contains the name of the file to write to and the second parameter is the string to be written.

Example 12.4: Writing to a File

```php
<?php
// 'w' mode overwrites the file. 'a' mode appends to the file.
$myfile = fopen("newfile.txt", "w") or die("Unable to open
file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Week 12 Homework

1. Create a simple "Contact Us" form with fields for Name, Email, and Message.

2. Write a PHP script that receives the data from this form using the POST method.
3. Implement server-side validation for the form:
    - Name must not be empty.
    - Email must be a valid email address.
    - Message must not be empty and should be at least 10 characters long.
4. If validation passes, sanitize the input and append the message to a file named `messages.txt`. Each entry should include the date, name, and email. For example: `[2026-01-19 14:30:00] John Doe (john.doe@email.com): This is my message.`
5. If validation fails, display the form again with the previously entered data and show error messages next to the invalid fields.

Week 13: Sessions, Cookies, and Database Connection

13.1 PHP Sessions

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user's computer. When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

A session is started with the `session_start()` function. Session variables are set with the PHP global variable: `$_SESSION`.

Example 13.1: Starting a Session and Setting Variables

```
<?php
// Start the session
session_start();

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

Example 13.2: Getting Session Variable Values

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

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

13.2 PHP Cookies

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user';s computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

A cookie is created with the `setcookie()` function. The first parameter is the name, the second is the value, and the third (optional) is the expiration date.

Example 13.3: Creating and Retrieving a Cookie

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
// Cookie expires in 30 days (86400 seconds = 1 day)
setcookie($cookie_name, $cookie_value, time() + (86400 * 30),
"/");
?>
<html>
```

```
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

To delete a cookie, you must set the cookie with a date that has already passed.

```
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
```

13.3 Database Connection with MySQLi

PHP 5 and later can work with a MySQL database using MySQLi (MySQL Improved) or PDO (PHP Data Objects). MySQLi is an extension that provides a procedural and an object-oriented interface to MySQL databases.

To connect to a MySQL database, you need the server name, username, password, and database name.

Example 13.4: MySQLi Connection (Object-Oriented Style)

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

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);
```

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

// Close connection
$conn->close();
?>
```

Example 13.5: MySQLi Connection (Procedural Style)

```
<?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_connect_error());
}
echo "Connected successfully";

// Close connection
mysqli_close($conn);
?>
```

Week 13 Homework

1. **User Login System:**
    - Create a simple login form (username and password).
    - On submission, check if the username is "admin" and the password is "password123".

- If the credentials are correct, start a session, set a session variable like `$_SESSION['user'] = 'admin';`, and redirect to a "welcome.php" page.
- On "welcome.php", check if the session variable is set. If it is, display "Welcome, [username]!". If not, redirect back to the login page.
- Add a "Logout" link on "welcome.php" that destroys the session and redirects to the login page.

2. **Visit Counter:**
   - Create a PHP script that uses a cookie to count the number of times a user has visited the page.
   - On the first visit, it should display "Welcome! This is your first visit."
   - On subsequent visits, it should display "Welcome back! You have visited this page X times." (where X is the visit count).
   - The cookie should expire in one year.
3. **Database Connection Test:**
   - Using phpMyAdmin or another tool, create a new database named `my_school`.
   - Inside `my_school`, create a table named `courses` with three columns: `id` (INT, AUTO_INCREMENT, PRIMARY KEY), `course_name` (VARCHAR), and `course_code` (VARCHAR).
   - Write a PHP script that connects to the `my_school` database. If the connection is successful, it should print "Successfully connected to the my_school database!". If it fails, it should print a detailed error message.

Week 14: CRUD Operations and Security

14.1 CRUD: Create, Read, Update, Delete

CRUD operations are the four fundamental functions of persistent storage. They are the building blocks for most web applications that interact with a database.

- **Create:** Adding new data records (e.g., a new user, a new blog post). This is done with the SQL `INSERT` statement.
- **Read:** Retrieving data from the database (e.g., displaying a list of products, viewing a single user's profile). This is done with the SQL `SELECT` statement.
- **Update:** Modifying existing data records (e.g., changing a user's email address). This is done with the SQL `UPDATE` statement.
- **Delete:** Removing data records from the database (e.g., deleting a comment). This is done with the SQL `DELETE` statement.

14.2 Prepared Statements for Security

When executing SQL queries with user-provided data, it is critical to use **prepared statements**. This technique prevents SQL injection attacks, one of the most common web hacking techniques. Prepared statements separate the SQL command from the data, ensuring that user input is treated as data only, not as executable code.

> ### Example 14.1: CRUD with Prepared Statements (MySQLi)
>
> Let's assume we have a `products` table with `id`, `name`, and `price` columns.

```
<?php
// Database connection ($conn) is assumed to be established

// --- CREATE (INSERT) ---
```

```
$stmt = $conn->prepare("INSERT INTO products (name, price)
VALUES (?, ?)");
// "sd" means the parameters are one string and one double.
$stmt->bind_param("sd", $name, $price);

// Set parameters and execute
$name = "Laptop";
$price = 1200.50;
$stmt->execute();

$name = "Mouse";
$price = 25.00;
$stmt->execute();

echo "New records created successfully<br>";
$stmt->close();



// --- READ (SELECT) ---
$sql = "SELECT id, name, price FROM products";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table border='1'><tr><th>ID</th><th>Name</th>
<th>Price</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>".$row["name"]."
</td><td>".$row["price"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
echo "<br>";



// --- UPDATE ---
$stmt = $conn->prepare("UPDATE products SET price = ? WHERE
name = ?");
$stmt->bind_param("ds", $new_price, $product_name);

// Set parameters and execute
$new_price = 1150.00;
$product_name = "Laptop";
$stmt->execute();
```

```
echo "Record updated successfully<br>";
$stmt->close();


// --- DELETE ---
$stmt = $conn->prepare("DELETE FROM products WHERE name = ?");
$stmt->bind_param("s", $product_name_to_delete);

// Set parameters and execute
$product_name_to_delete = "Mouse";
$stmt->execute();

echo "Record deleted successfully<br>";
$stmt->close();

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

14.3 Advanced Security Practices

Password Hashing

Never store passwords in plain text. Always use a strong, one-way hashing algorithm. PHP's built-in `password_hash()` and `password_verify()` functions are the recommended way to handle passwords.

Example 14.2: Password Hashing

```
<?php
// Hashing a password for storage
$password = 'mysecretpassword';
$hash = password_hash($password, PASSWORD_DEFAULT);

// Now, you would store $hash in your database, not $password

// Verifying a password upon login
$login_password = 'mysecretpassword';
if (password_verify($login_password, $hash)) {
    echo 'Password is valid!';
} else {
    echo 'Invalid password.';
```

```
}
?>
```

Preventing Cross-Site Scripting (XSS)

XSS attacks occur when malicious scripts are injected into otherwise benign and trusted websites. The primary defense is to escape all output. Use the `htmlspecialchars()` function whenever you display user-provided data.

Example 14.3: XSS Prevention

```
<?php
// Assume this comes from a user form
$comment = '<script>alert("XSS attack!");</script>';

// Displaying it safely
echo htmlspecialchars($comment, ENT_QUOTES, 'UTF-8');
// Output will be the literal text, not an executable script.
?>
```

Preventing Cross-Site Request Forgery (CSRF)

CSRF attacks trick a user into performing an unwanted action on a web application in which they';re authenticated. The defense is to use anti-CSRF tokens. A unique, secret, unpredictable token is generated for each user session and embedded in forms. The server then checks for this token upon submission.

Example 14.4: CSRF Token Implementation

```
<?php
session_start();
// Generate a token if one doesn't exist
if (empty($_SESSION['token'])) {
    $_SESSION['token'] = bin2hex(random_bytes(32));
}
$token = $_SESSION['token'];
?>
```

```
<!-- In your form -->
<form action="process.php" method="post">
    <input type="hidden" name="csrf_token" value="<?php echo
$token; ?>">
    <!-- other form fields -->
    <input type="submit" value="Submit">
</form>

<?php
// In process.php
session_start();
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!hash_equals($_SESSION['token'],
$_POST['csrf_token'])) {
        // Token does not match, handle error
        die('CSRF token validation failed!');
    } else {
        // Process the form
        echo 'Form processed successfully!';
    }
}
?>
```

Week 14 Homework

1. **Create a "To-Do List" Application:**
    - Create a database table named `tasks` with columns: `id`, `task_description` (TEXT), `is_completed` (BOOLEAN/TINYINT), `created_at` (TIMESTAMP).
    - **Create:** Build a form to add new tasks to the database. Use prepared statements.
    - **Read:** Display all tasks from the database. Show completed tasks with a line-through style.
    - **Update:** Add a checkbox next to each task. When a user checks/unchecks it, update the `is_completed` status in the database.
    - **Delete:** Add a "Delete" button for each task that removes it from the database.

- **Security:** Ensure all user input is sanitized with `htmlspecialchars()` and all database queries use prepared statements.

Week 15: Object-Oriented PHP and Final Project

15.1 Object-Oriented Programming (OOP) in PHP

Object-Oriented Programming is a paradigm based on the concept of "objects", which can contain data in the form of fields (often known as attributes or properties) and code in the form of procedures (often known as methods). OOP helps in creating modular, reusable, and maintainable code.

Classes and Objects

A **class** is a template for creating objects. It defines properties (variables) and methods (functions) that the objects will have. An **object** is an instance of a class.

Example 15.1: A Simple `Car` Class

```
<?php
class Car {
    // Properties
    public $color;
    public $model;

    // Constructor
    public function __construct($color, $model) {
        $this->color = $color;
        $this->model = $model;
    }

    // Method
    public function getDetails() {
        return "This is a " . $this->color . " " . $this-
>model . ".";
    }
```

```
}

// Create an object (instance) of the Car class
$myCar = new Car("Red", "Toyota Camry");

// Call the method
echo $myCar->getDetails(); // Outputs: This is a Red Toyota
Camry.
?>
```

Inheritance

Inheritance is a mechanism where a new class (child class) inherits properties and methods from an existing class (parent class). This promotes code reuse.

Example 15.2: Inheritance

```
<?php
// Parent class
class Animal {
    public $name;

    public function __construct($name) {
        $this->name = $name;
    }

    public function eat() {
        echo $this->name . " is eating.<br>";
    }
}

// Child class
class Dog extends Animal {
    public function bark() {
        echo $this->name . " says Woof!<br>";
    }
}

$myDog = new Dog("Buddy");
$myDog->eat();  // Inherited from Animal class
```

```
$myDog->bark(); // Method from Dog class
?>
```

Access Modifiers: `public`, `protected`, `private`

- **public:** The property or method can be accessed from anywhere. This is the default.
- **protected:** The property or method can be accessed within the class and by classes derived from that class.
- **private:** The property or method can ONLY be accessed within the class that defines it.

15.2 Final Project Work Session

This week is dedicated to working on your final project. The goal is to apply all the concepts learned throughout the 15-week course to build a complete, functional, and secure web application.

Project Ideas Review:

- Student Management System
- Blog Platform
- E-Commerce Store
- Task Management System
- Library Management System

Key Requirements Checklist:

- [ ] User Authentication (Login/Registration) with Password Hashing
- [ ] Database with at least 3 related tables
- [ ] Full CRUD (Create, Read, Update, Delete) functionality
- [ ] Server-side and Client-side Form Validation
- [ ] File Uploads (e.g., profile pictures, product images)
- [ ] Session Management (for logged-in users)
- [ ] Security: Use Prepared Statements, `htmlspecialchars()`, and CSRF tokens
- [ ] Clean, organized, and commented code

- [ ] Responsive and user-friendly interface

Project Structure Recommendation:

```
/project-root
|-- /css
|   `-- style.css
|-- /js
|   `-- script.js
|-- /images
|   `-- (uploaded images go here)
|-- /includes
|   |-- db_connect.php (database connection)
|   |-- functions.php (helper functions)
|   |-- header.php
|   `-- footer.php
|-- index.php (main page, e.g., student list)
|-- login.php
|-- logout.php
|-- register.php
|-- create.php (form to add new item)
|-- edit.php (form to edit an item)
|-- delete.php (script to delete an item)
|-- README.md
`-- database.sql (SQL dump of your database structure)
```

15.3 Final Project Submission

Your final project is the culmination of this course. It should be a polished application that demonstrates your understanding of web development principles.

Submission Guidelines:

- **Code:** A ZIP file containing your entire project folder.
- **Database:** An SQL file (`database.sql`) that contains the `CREATE TABLE` statements for your database.
- **README.md:** A markdown file that includes:
  - Project title and a brief description.
  - Instructions on how to set up and run the project (e.g., database configuration).
  - A list of features you implemented.
  - Credentials for a test user account.

Week 15 Homework

Complete and submit your final web application project based on the requirements outlined above. Ensure your code is clean, well-commented, and secure. This project will be a significant part of your final grade and will serve as a portfolio piece to showcase your skills to potential employers.

Deadline: End of Week 15.

Good luck, and happy coding!

References

1. Robbins, J. N. (2012). *Learning Web Design* (4th ed.). O'Reilly.
2. Duckett, J. (2011). *HTML & CSS: Design and Build Websites*. Wiley.
3. W3Schools.com - Online Web Tutorials.
4. PHP.net - Official PHP Documentation.
5. PHP: The Right Way - Best practices guide.