Certificate in Computer Applications (CCA)
Study Material

CCA - 104 Web Technologies Part 2
(Unit 4 to Unit 5)

Supported by
Institute of Management Studies (IMS),
Ghaziabad-UP

About CCA Program

The certificate program focuses on computer fundamentals. This program provides a comprehensive introduction to Fundamentals of Information Technology; Computer Applications; Internet & Communication Technologies; Web Programming; and Soft Skills.

The program is designed and conducted by CSC Academy along with one of the leading Management Institute, Institute of Management Studies, Ghaziabad (UP). Some of the core subject faculty are associated in delivering this program.

After the completion of this course, student will be able to:

- Get a basic understanding of personal computers and their operations.
- Use of MS Office Tools - Like MS word, MS excel and Power point presentations
- Understand basics of Programming.
- Recognize and describe the working of Computer Networks.
- Get familiar with the basics of communication skills
- Develop good skills at writing business letters, emails, minutes of meeting and other business correspondence.
- Design and Implement interactive, responsive web site using HTM5L, CSS5 and JavaScript.
- Build Dynamic web site using server-side PHP Programming and Database connectivity.

The CCA program covers five course modules:

Unit 101: Fundamentals of IT & Programming

Unit 102: Data Communications

Unit 103: Soft Skills & Communications

Unit 104: Web Technologies

Unit 105: Cyber Security

The objective of this study material is to provide the students to enable them to obtain knowledge and skills in the related subject. This material is not in itself to be read alone, and student should use this in addition to the CCA online e-learning content study. In case students need any further clarifications or have any suggestions to make for further improvement of the material contained herein, they may give the same at CSC Academy Centre.

All care has been taken to provide content in a manner useful to the students.

Permission of the CSC Academy is essential for reproduction of any portion of this material.

Edition : June 2020

Published by : CSC Academy

About CSC Academy

CSC Academy was setup in 2017 that provides access to professional learning for learners of diverse backgrounds and educational needs. The CSC Academy is a not-for-profit society under the Societies Registration Act 1860 (Act 21 of 1860), as applicable to the Union of Delhi with its registered office in Delhi. The CSC Academy board comprises of the Additional Secretary, Ministry of Electronics & Information Technology, Government of India as Chairman, and others reputed members from academia. CSC Academy has received certificate from Income Tax Department under section 12 AA and 80 G.

The CSC Academy is committed to teaching, delivering of specialized courses/ training programs, leadership, communication skills and promotion of entrepreneurship among the rural masses in India. Presently, the CSC Academy is delivering various Government of India sponsored skill and education programs, in addition to courses from private sector.

About Institute of Management Studies, Ghaziabad (UP)

IMS Ghaziabad is a pioneer institute for management education in Northern India. It is the first institute of IMS Society Ghaziabad with 30 glorious years of excellence. IMS Ghaziabad offers full time AICTE approved & NBA accredited PGDM Programme which has been awarded the MBA equivalent status by the Association of Indian Universities (AIU), PGDM - International Business, PGDM - Big Data Analytics and MCA Programme are approved by AICTE and affiliated to Dr APJ AKTU, Lucknow.

Since its foundation IMS Ghaziabad has gathered a lot of feathers in its cap with global accreditations and memberships such as Accreditation Services for International Colleges (U.K), AACSB Business Education Alliance, National Assessment and Accreditation Council - 'A' Grade.

IMS Ghaziabad is amongst Top 10 best B-Schools in North India as per latest MBA and B School Rankings. It has been awarded as the "Best Campus for Industry Oriented Management Education in India / Asia Pacific 2019" by ASSOCHAM and the Education Post. It has been ranked as 5th in North India and 15th in India by Times of India B School Survey, February 2019, A++ Institute in Delhi - NCR by 9th Chronicle B-School Survey 2018.

Table of Contents

Course Outline

Course Objective

To familiarize with basics of the Internet programming and acquire knowledge and skills for creation of web site considering both client and server-side programming. It emphasizes to gain ability to develop responsive web applications.

Course Outcomes

At the end of this course, student should be able to:

1. Implement interactive web page(s) using HTML, CSS and JavaScript.
2. Design a responsive web site using HTML5 and CSS3.
3. Demonstrate Rich Inter Application.
4. Build Dynamic web site using server-side PHP Programming and Database connectivity.
5. Describe and differentiate different Web Extensions and Web Services.

Course Outline

Unit I Designing and Planning Web Pages

Designing and Planning Web Pages, Surveying the Possibilities, Website Evaluation Tool, Color Theory in Web Designing, Selecting a Color Scheme, Web Standards & Accessible Design, How People with Disabilities Access the Web, Planning & Organizing a Website.

UNIT II Introduction to HTML

Creating Pages with HTML, Basic HTML Markup, Basic Elements of HTML, Essential Tags, Common Tags, HTML Lists, Unordered Lists, Ordered Lists, Nested Lists, Creating Links, Linking to External Internet Sites, Creating a Data Table, Formatting Web Pages with Style Sheets, Introduction to Cascading Style Sheets, Anatomy of a Style, Applying Styles, Applying Styles to Data Tables, Page Layout Techniques, Layout with CSS, Layout with Tables, Using an external style sheet, Linking to an External Style Sheet.

UNIT III Introduction to Web Graphics

Introduction to Web Graphics, Copyright Law and Graphics on the Web, Understanding Web Graphics, Acquiring Images for Web Graphics, Cropping and Resizing, Adding Images to Your Web Page, Creating Navigation Buttons, Basic Shapes and Colors, Working With Text, Layer Basics, Basic

Image Manipulation, Selection Tools, Layer Effects.

UNIT IV Scripting (JavaScript & CSS)

A Simple JavaScript Program, Validating a Website, Validating Your HTML, Validating Your CSS, Introduction to Web Authoring Software, Basic Features of Web Authoring Software, Constructing the Client Website.

UNIT V Introduction to Server, Database and PHP

Introduction to Server, Introduction to Server-Side Scripting, Database, Sever installation & Working (WAMP), Introduction to PHP, PHP Script designing, Registration & Login Page, Maintenance & Handling of Server, Creating contact us page, Database Creation & handling SQL Queries, Live Running of Website.

Reference books

1. Jeffrey C. Jackson, “Web Technologies-A Computer Science Perspective”, Pearson Education 2006.
2. Web Technologies, Black Book, Dreamtech Press.
3. Web Applications: Concepts and Real World Design, Knuckles, Wiley-India.
4. Internet and World Wide Web How to Program, P. J. Deitel & H M Deitel, Pearson.
5. Learning PHP, MySQL and JavaScript, Robin Nixon, O’Reilly.

Unit 4 Scripting (JavaScript & CSS)

Unit 4.1 JavaScript

Browser Scripting

- Browser scripting languages allow dynamic behavior to be specified within HTML documents.
- A scripting language is a lightweight programming language
- Browsers must support the used scripting language.
- Browsers are disabled for scripting to prevent the risk of misuse.

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages.
- JavaScript is a scripting language.
- A JavaScript consists of lines of executable computer code.
- A JavaScript is usually embedded directly into HTML pages.
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation).
- Everyone can use JavaScript without purchasing a license.

JavaScript

- **Are Java and JavaScript the Same?**
- NO!
- Java and JavaScript are two completely different languages in both concept and design!
- Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript Do?

- JavaScript gives HTML designers a programming tool.
- JavaScript can put dynamic text into an HTML page.
- JavaScript can react to event.
- JavaScript can read and write HTML elements.
- JavaScript can be used to validate data.
- JavaScript can be used to detect the visitor's browser.
- JavaScript can be used to create cookies.

How to Put a JavaScript Into an HTML Page

```
<html>
<body>
    <script type="text/javascript">
       document.write("Hello
  World!")
    </script>
</body>
</html>
```

Where to Put the JavaScript

- Head section
- Body section
- External scripts

JavaScript in the head section

```
<html>
<head>
   <script type="text/javascript">
      function message()
         {
              alert("This alert box was called with
             the onload event")
          }
   </script>
</head>

<body onload="message()">

</body>
</html>
```

Scripts in the body section

```
<html>
<head>
</head>
<body>
    <script type="text/javascript">
      document.write("Hello World!")
    </script>
</body>
```

Using External JavaScript

```
<html>
<head>
   <script src="xyz.js">
   </script>
</head>
<body>
</body>
</html>
```

What can a JavaScript Do?

```
<html>
<body>
   <script type="text/javascript">
      var d = new Date()
      var time = d.getHours()

       if (time < 10)
          {
             document.write("<b>Good morning</b>")
          }
        else
          {
             document.write("<b>Good day</b>")
          }
   </script>
```

…

What can a JavaScript Do?

```
<html>
<body>

   <script type="text/javascript">
      for (i = 0; i <= 5; i++)
         { document.write("The number is " + i)
           document.write("<br />")
         }
   </script>
<p>Explanation:</p>
<p>This for loop starts with i=0.</p>
<p>As long as <b>i</b> is less than, or equal to 5, the loop
   will continue to run.</p>
<p><b>i</b> will increase by 1 each time the loop runs.</p>

</body>
</html>
```

JavaScript Objects

- Window: The top level object in the JavaScript hierarchy. The Window object represents a browser window. A Window object is created automatically with every instance of a <body> or <frameset> tag
- Navigator: Contains information about the client's browser
- Screen: Contains information about the client's display screen
- History: Contains the visited URLs in the browser window
- Location: Contains information about the current URL

DOM Objects

- DOM Anchor
  DOM Base
  DOM Button
  DOM Form
  DOM Frameset
  DOM Image
  DOM Input Checkbox
  DOM Input Hidden
  DOM Input Radio
  DOM Input Submit
  DOM Link
  DOM Object
  DOM Select
  DOM Table
  DOM TableRow
  DOM Area
  DOM Body
  DOM Event
  DOM Frame
  DOM IFrame
  DOM Input Button
  DOM Input File
  DOM Input Password
  DOM Input Reset
  DOM Input Text
  DOM Meta
  DOM Option
  DOM Style
  DOM TableCell
  DOM Textarea

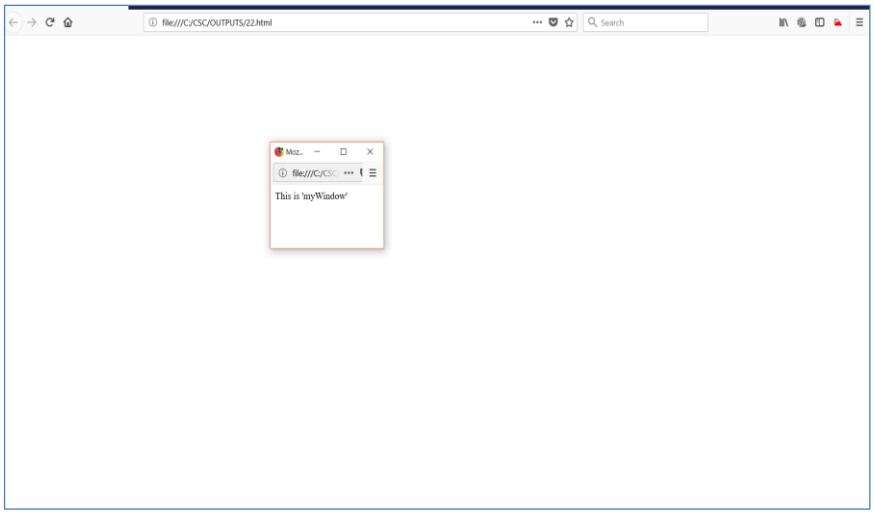
Window Example

```
<html>
<body>
  <script type="text/javascript">
    window.status="Some text in the
      status bar!!"
  </script>
</body>
</html>
```

Window Example

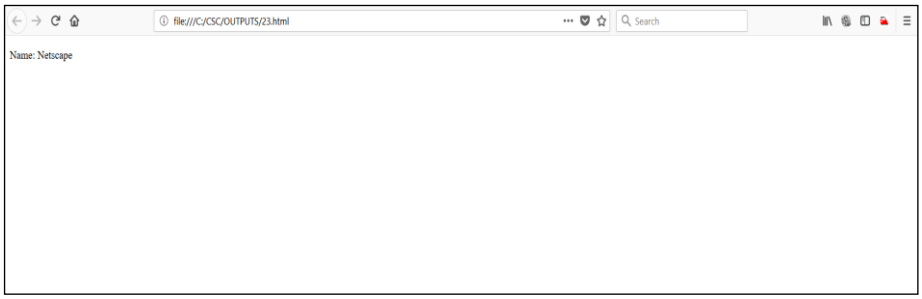
```
<html>
<body>
<script type="text/javascript">
  myWindow=window.open('','','width=200, height=100')
  myWindow.document.write("This is 'myWindow'")
  myWindow.focus()
</script>
</body>
</html>
```

Window Example



Browser Example

```
<html>
<body>
   <script type="text/javascript">
        document.write("<p>Name: ")
        document.write(navigator.appName + "</p>")
   </script>
</body>
</html>
```



Screen Example

```
<html>
<body>
   <script type="text/javascript">
        document.write("<p>Width: ")
        document.write(screen.width + "</p>")
   </script>
</body>
</html>
```

Anchor Example

```
<html>
<head>
<script type="text/javascript">
 function changeLink() {
  document.getElementById('myAnchor').innerHTML="W3Schools";
  document.getElementById('myAnchor').href="http://www.w3schools.com";
  document.getElementById('myAnchor').target="_blank"; }
</script>
</head>

<body>
 <a id="myAnchor" href="http://www.microsoft.com">Microsoft</a>
 <input type="button" onclick="changeLink()" value="Change link">
</body>
</html>
```

Table Example

```
  <head>
  <script type="text/javascript">
     function changevalue()     {
          var x=document.getElementById('mytable').cells
                 x[0].innerHTML="Ali"
                 x[1].innerHTML="Veli"
                 x[2].innerHTML="Selami"       }
  </script>
  </head>

  <body>
  <table id="mytable" border=1>
    <tr>
      <td id="col1"> John </td>
      <td id="col2"> Doe </td>
      <td id="col3"> Alaska </td>
    </tr>
    </table>
  <input type="button" value="Change value" onclick="changevalue()">
  ……
```

Unit 4.2 : Using Style Sheets - CSS

What is CSS?

- A simple mechanism for controlling the style of a Web document without compromising its structure.
- It allows you to separate visual design elements (layout, fonts, colors, margins, and so on) from the contents of a Web page.
- Allows for faster downloads, streamlined site maintenance, and global control of design attributes across multiple pages.

2

CSS vs just HTML

What can we do with CSS that we can't do with HTML?

- Control of backgrounds.
- Set font size to the exact height you want.
- Highlight words, entire paragraphs, headings or even individual letters with background colors.
- Overlap words and make logo-type headers without making images.
- Precise positioning.
- Linked style sheets to control the look of a whole website from one single location.
- And more.

3

The Anatomy of Cascading Style Sheets (CSS)

- **Selector**
  - HTML element tags
    (examples: p, h2, body, img, table)
  - class and ID names
- **Property** (examples: color, font-size)
- **Value** (examples: red, 14pt)

4

The Anatomy of Cascading Style Sheets (CSS)

- The basic syntax of a CSS rule:
  selector {property 1: value 1; property 2: value 2}

  Example:
  p {font-size: 8pt; color: red}

  Notice the **{ }** around the rule and the **:** before each value!

5

Three ways to include CSS:

1. Local (Inline)
2. Global (Embedded, or Internal)
3. Linked (External)

6

1. Local

- **Inline** style sheet.
- Placed inside tags.
- Specific to a single instance of an html tag on a page.
- **Must** be used instead of <font> tags to specify font size, color, and typeface and to define margins, etc.
- Use to override an external or embedded style specification.

7

Local (inline)

- Example

```
<p style="font-size: 10pt; color: red; font-weight: bold;
font-family: Arial, Helvetica, sans-serif">
    This is a local stylesheet declaration. </p>
```

On the browser: ➡ **This is a local stylesheet declaration.**

8

2. Global

- **Embedded** or **internal** style sheet
- Applicable to an entire document
- Styles are defined within the <style> </style> tag, which is placed in the **header** of the html file (i.e., within <head> and </head>).

9

Global (Internal)

- Example:

```
<html>
  <head>
   <title>Title</title>
    <style type="text/css">
      <!--[STYLE INFORMATION GOES HERE] -->
    </style>
  </head>
 <body>
      [DOCUMENT BODY GOES HERE]
  </body>
</html>
```

10

3. Linked

- **External** style sheet
- Styles are saved in a separate file, with the extension **.css**
- This single stylesheet can be used to define the look of multiple pages.

11

Linked (External)

- Example

In TextPad,Notepad, etc....

```
p {font-family: verdana, sans-serif;
font-size: 12pt; color: red}

h1 {font-family: serif; font-size:
14pt; color: green}

h2 {font-family: serif; font-size:
11pt; color: blue}
```

Save this text file as *whatever.css*

12

Linked (External)

- Example (continued)

To apply the stylesheet "***whatever.css***" to an HTML document, call it in from the header:

```
<head>
  <link rel="stylesheet"
  href="whatever.css" type="text/css">
</head>
```

13

Applying Styles to Data Tables

Up to this point, as we haven't applied any CSS styling to our tables, our example tables have not been too pleasing to the eye. CSS offers us several properties to customize how our tables appear on the page:

Style	Description
width	Width of element.
background-color	Background color of element.
color	Color of text in element.
text-align	Horizontal text alignment of element.
border	Border thickness and style of element.
padding	Padding (white space around content) of element.

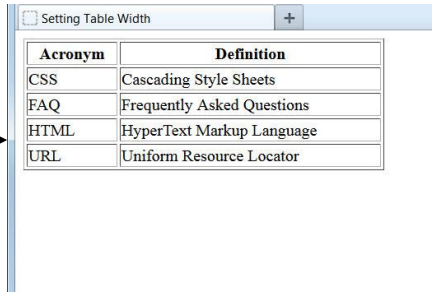
We're already familiar with the **color** and **text-align** properties, but let's see examples of the other styles in action.

Though we're using these properties to style table elements, all of these properties may be used with many other XHTML elements, as we'll see in future lessons.

Setting Table Width

We can set the overall width of a table by applying a class to the <table> element:

```
<style type="text/css">
  .glossary {
    width: 350px;
  }
</style>
...
<table class="glossary" border="1">
  <tr>
    <th>Acronym</th>
    <th>Definition</th>
  </tr>
  <tr>
    <td>CSS</td>
    <td>Cascading Style Sheets</td>
  </tr>
  ...
</table>
```



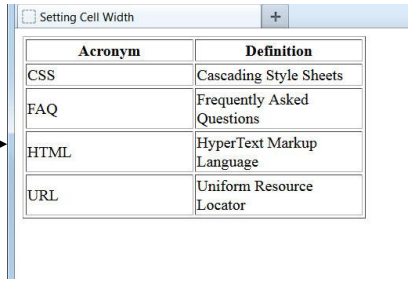
The table is now 350 pixels wide, creating some extra space in the data cells.

The width of the columns is automatically established by the browser to accommodate the cell contents, but we don't have to accept this format.

Setting Column Width in Pixels

We can set the width of columns by setting the widths of the first row of data cells:

```
<style type="text/css">
  .glossary {
    width: 350px;
  }
  .col {
    width: 175px;
  }
</style>
...
<table class="glossary" border="1">
  <tr>
    <th class="col">Acronym</td>
    <th class="col">Definition</td>
  </tr>
  <tr>
    <td>CSS</td>
    <td>Cascading Style Sheets</td>
  </tr>
  ...
</table>
```



Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

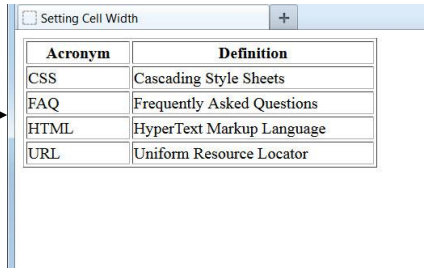
We've now set each of the columns to be 175 pixels in width, or half of the total available table width.

Notice that due to the enforced width, some cell contents had to wrap to an additional line.

Setting Column Width in Percent

Rather than pixels, we can also set column widths to be a percent of the total available width:

```
<style type="text/css">
  .glossary {
    width: 350px;
  }
  .col1 {
    width: 30%;
  }
  .col2 {
    width: 70%;
  }
</style>
...
<table class="glossary" border="1">
  <tr>
    <th class="col1">Acronym</td>
    <th class="col2">Definition</td>
  </tr>
  ...
</table>
```



Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Setting column widths by percent instead of pixels has the advantage of flexibility. Should we ever alter the width of the table itself, the columns will automatically adjust.

Borders

CSS provides us plenty of flexibility when generating borders around elements. There are multiple properties that affect the border that displays on the page:

Style	Description
border-style	Type of border: none, solid, dashed, dotted, double, groove, ridge, inset, outset.
border-color	Color of border.
border-width	Width of border, measured in pixels. Also available: thin, medium, and thick.
border-collapse	collapse: borders display as single border. separate: borders are detached (default).

We'll examine each of the first three, but first let's take a look at the **border-collapse** property.

The border-collapse Property

By invoking the **border-collapse** property, we can force a table to collapse its borders into a single line between cells:

```
<style type="text/css">
  .glossary {
    width: 350px;
  }
  .collapse {
    border-collapse: collapse;
  }
...
</style>
...
<table class="glossary collapse"
   border="1">
  <tr>
    <th class="col1">Acronym</td>
    <th class="col2">Definition</td>
  </tr>
...
</table>
```

Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

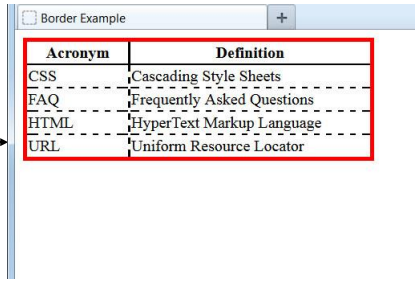
Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Here we created an identical table but added a class that included the **border-collapse** property.

Customizing Borders

By using the other three properties, we can create custom borders for our table:

```
<style type="text/css">
  .glossary {
    width: 350px;
    border-width: 4px;
    border-style: solid;
    border-color: red;
  }
  th {
    border-width: 2px;
    border-style: solid;
  }
  td {
    border-width: 2px;
    border-style: dashed;
  }
  ...
```



Acronym	Definition
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Notice that we can apply different styles to the table, table headers, and table cell elements.

For demonstrative purposes, we applied CSS styles directly to the <th> and <td> elements, instead of using separate classes. In the real world, this would likely be a bad idea, as any other tables on our page would be affected too.

Using CSS Border Shorthand

When specifying multiple border properties, we can use the CSS **border shorthand** to reduce the statement to a single line:

```
<style type="text/css">
  .glossary {
    width: 350px;
    border-width: 4px;
    border-style: solid;
    border-color: red;
  }
  th {
    border-width: 2px;
    border-style: solid;
  }
  td {
    border-width: 2px;
    border-style: dashed;
  }
  ...
```

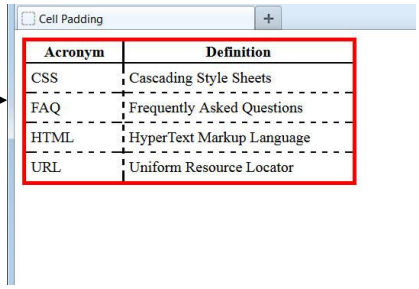
```
<style type="text/css">
  .glossary {
    width: 350px;
    border: 4px solid red;
  }
  th {
    border: 2px solid;
  }
  td {
    border: 2px dashed;
  }
  ...
```

By convention, the properties are ordered as **border: border-width border-style border-color;** (The color portion may be omitted.)

Adding Cell Padding

By setting the **padding** property, we can make sure there is at least that much white space around our cell contents. This keeps the actual cell contents from displaying too closely to the borders:

```
<style type="text/css">
  .glossary {
    width: 350px;
    border: 4px solid red;
  }
  th {
    border: 2px solid;
  }
  td {
    border: 2px dashed;
    padding: 5px;
  }
...
```

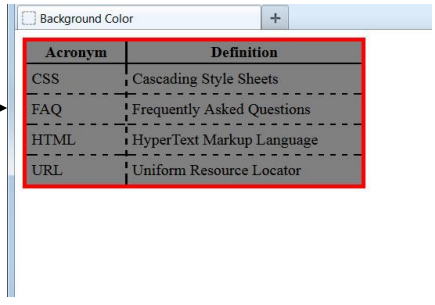


Different border and padding settings can be set for the top, bottom, left, and right sides of elements. We will learn how to do this in an upcoming lesson.

Setting a Background Color

By setting the **background-color** property, we can change our table's background away from the default:

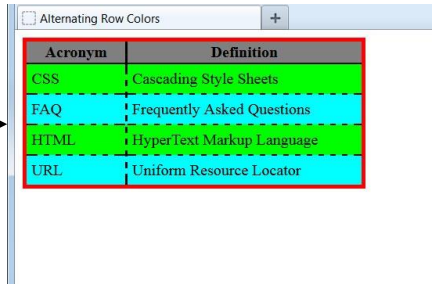
```
<style type="text/css">
  .glossary {
    width: 350px;
    border: 4px solid red;
    background-color: gray;
  }
  th {
    border: 2px solid;
  }
  td {
    border: 2px dashed;
    padding: 5px;
  }
...
```



Let's use this property now on the data cells to make our table a bit more readable.

Using Background Color on Rows

```
<style type="text/css">
...
  .odd {
    background-color: lime;
  }
  .even {
    background-color: aqua;
  }
...
</style>
...
  <tr class="odd">
    <td>CSS</td>
    <td>Cascading Style Sheets</td>
  </tr>
  <tr class="even">
    <td>FAQ</td>
    <td>Frequently Asked Questions</td>
  </tr>
  <tr class="odd">
...
```



This is a handy way to make our table rows show in alternating background colors. This technique makes reading wide tables far easier for viewers.

Using an external style sheet and Linking to External Style Sheet

External style sheets are created in separate documents with a .css extension. An external style sheet is simply a listing of CSS rules. It cannot contain HTML tags. The <link> tag, which goes in the head of an HTML page, is used to link to an external style sheet. There is no limit to the number of external style sheets a single HTML page can use. Also, external style sheets can be combined with embedded style sheets. Follow these steps to create an external style sheet.

1. Start with an HTML file that contains an embedded style sheet, such as this one. Copy this text and paste into a new HTML file

```
<html>
<head>
<meta charset="UTF-8">
<title>Embedded Style Sheet</title>
<style type="text/css">
h1 { text-align: center; font-size:
    12pt; color: #000099;
    margin-bottom: }
table { margin: 5px; width: 290px; }
th { padding: 3px; }
td { padding-left: 8px;
    padding-right: 8px;
    border: 1px solid #990000}
#trHeader { text-decoration: underline;
    color: #990000; }
.centerCell { text-align: center; }
</style> </head> <body>
<div>
<h1>All-time Home Run Record</h1>
<table>
<tr id="trHeader">
            <th>Player</th>
            <th>Home Runs</th>
            <th>Team</th>
</tr>
<tr>
<td>Barry Bonds</td>
<td class="centerCell">762</td>
<td>Giants</td>
</tr>
<tr>
<td>Hank Aaron</td>
<td class="centerCell">755</td>
<td>Braves</td>
 </tr>
</table> </div> </body> </html>
```

2. Create a new file and save it as StyleSheet.css in the same directory. (You can give the file any name as long as it has the .css extension).

3. Move all the CSS rules from the HTML file to the StyleSheet.css file. Don't copy the style tags.

```
h1 { text-align: center; font-size:
    12pt; color: #000099;
    margin-bottom: }
table { margin: 5px; width: 290px; }
th { padding: 3px; }
td { padding-left: 8px;
    padding-right: 8px;
    border: 1px solid #990000}
#trHeader { text-decoration: underline;
    color: #990000; }
.centerCell { text-align: center; }
```

4. Remove the style block from the HTML file.
5. In the HTML file, add a link tag after the closing title tag that points to StyleSheet.css.

<link href="StyleSheet.css" rel="stylesheet" type="text/css">

<link> attributes include:

- href: points to the location of the external style sheet
- rel: must be set to "stylesheet" for linking style sheets
- type: must be set to "text/css" for linking to cascading style sheets

Page Layout Techniques

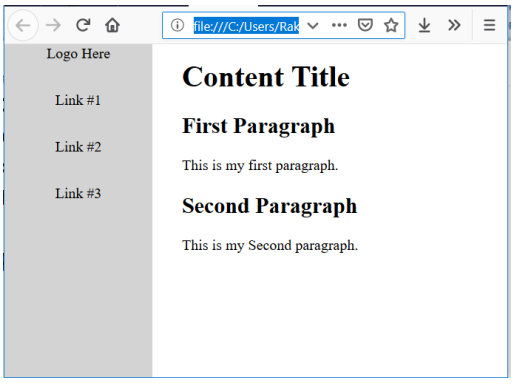
1. Two-column Page Layout
2. Sidebar Navigation Layout with Position
3. Flexbox Layout
4. Grid Layout

Two-column Page Layout

```
<!DOCTYPE html>
<html>
<head>
  <title>2 Columns</title>
  <style type="text/css">
   div:nth-of-type(1) {
       width: 49%; float: left; }
   div:nth-of-type(2) {
        width: 49%; float: right; }
  </style>
</head>
<body>
  <h1>2 Columns</h1>
  <div>
    <h2>First Column</h2>
    <p>This is my First Page</p>
  </div>
  <div>
    <h2>Second Column</h2>
    <p>This is my Secopnd Page</p>
  </div>
</body>
</html>
```

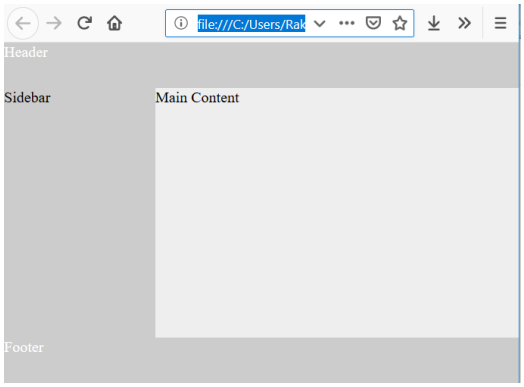
Sidebar Navigation Layout with Position

```
<html> <head>
<title>Sidebar Navigation</title>
<style>
 body, h1 { margin: 0; }
.navigation {
position: absolute; left: 0;
width: 10rem; height: 100%;
background-color: lightgray;
text-align: center; }
 .navigation ul { padding: 0;
 list-style-type: none; }
 .navigation li {
  padding: 1rem; }
  .content { padding-top: 1rem;
   margin-left: 12rem; }
</style>
</head>
<body>
<nav class="navigation">
<span>Logo Here</span>
<ul>
   <li>Link #1</li>
   <li>Link #2</li>
    <li>Link #3</li>
</ul> </nav>
<div class="content">
<h1>Content Title</h1>
<div>
<h2>First Paragraph</h2>
<p>This is my first paragraph.</p>
 </div>
<div>
<h2>Second Paragraph</h2>
<p>This is my Second paragraph.</p>
</div>
</div>
</body>
</html>
```



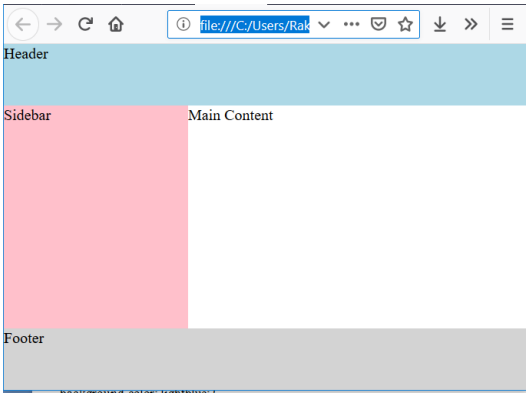
Flexbox Layout

```
<!DOCTYPE html>
<html>
<head>
 <title>Flexbox Layout</title>
<style>
body { margin: 0; min-height:
100vh; background: #ccc;
display: flex; flex-direction:
column; }
.header, .footer { height: 3rem;
background: #777;
color: white; }
.content { display: flex; flex: 1;
background: #999; color: #000; }
.main { flex: 1; background: #eee; }
.sidebar { width: 10rem; background:
#ccc; }
</style>
</head>
<body>
<header class="header">Header</header>
<main class="content">
<aside class="sidebar">Sidebar</aside>
<section class="main">Main Content</section>
</main>
<footer class="footer">Footer</footer>
</body> </html>
```



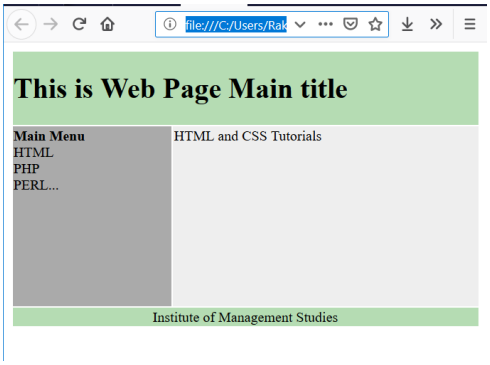
Grid Layout

```
<!DOCTYPE html>
<html>
<head>
 <title>Flexbox Layout</title>
<style>
html, body, .container {
width: 100%;height: 100%;margin: 0;}
.container {display: grid;
grid-template-columns: 12rem 1fr;
grid-template-rows: 4rem 1fr 4rem;
grid-template-areas: "header header“
 "sidebar content“"footer footer“;}
.header { grid-area: header;
 background-color: lightblue; }
.sidebar { grid-area: sidebar;
 background-color: pink;}
.content { grid-area: content;
background-color: white;}
.footer { grid-area: footer;
background-color: lightgray;}</style>
</head>
<body>
<div class="container">
   <div class="header">Header</div>
   <div class="sidebar">Sidebar</div>
   <div class="content">Main Content</div>
   <div class="footer">Footer</div>
</div>
</body> </html>
```



Layout with Table

```
<!DOCTYPE html>
<html>
 <head>
    <title>HTML Layout using Tables</title>
 </head>
<body>
  <table width = "100%" border = "0">
  <tr>
     <td colspan = "2" bgcolor = "#b5dcb3">
     <h1>This is Web Page Main title</h1></td>
  </tr>
  <tr valign = "top">
    <td bgcolor = "#aaa" width = "50">
    <b>Main Menu</b><br />HTML<br />
    PHP<br /> PERL…</td>
   <td bgcolor = "#eee" width = "100" height = "200">
   HTML and CSS Tutorials </td>
   </tr>
   <tr>
     <td colspan = "2" bgcolor = "#b5dcb3">
     <center>
          Institute of Management Studies
     </center>
     </td>
   </tr>
</table>
</body>
</html>
```



Unit 5 Server Side Scripting

Client Side vs. Server Side Web

- Simply defined, client-side code executes on the end-user's computer, usually within a web browser.
- Server-side code executes on the web server, usually within a web application environment, which in turn generates HTML to be viewed in a browser.

Client Side vs. Server Side Web

- Which one to choose? What are the determining factors?
  - Performance:
    - Responsiveness, speed, reliability
    - Ability to handle a large number of simultaneous users
  - Functionality:
    - Simplicity of use and maintenance,
    - Breadth of user options
    - Ability to handle multiple simultaneous transactions
  - Security:
    - Desktop security
    - Server security
    - Database security
    - Network security

Client Side vs. Server Side Web

- Examples:
  - Code that runs on the server that interprets every mouse move and keystroke is clearly undesirable
    - terminal to mainframe paradigm
  - On the other hand, one does not want to download an entire product database to a browser and then run code that searches for the products.
  - Server side forms have direct access to active code and perform more reliably
  - On the other hand they are more prone to slowdowns due to the server/network congestion

Client Side vs. Server Side Web

In general, the key areas where client-side coding has advantages stem from its location on the user desktop and/or other end device. They include the following:

- **Interactivity (e.g., mouse and keyboard handling)**
- **Handling of user interface controls: buttons, textboxes, etc.**
- **Feedback and validation**

Key server-side strengths include stem from their proximity to the backend business databases and other applications. They include the following:

- **Direct information access, retrieval, processing and storage**
  - **facilitate e-commerce, reservations, shipment tracking etc.**
- **central repository of added web features such as e-mail, chat and multimedia streaming**
- **security and authentication (mostly)**

Client Side Technologies

- Java Applets:
  - small programs written in Java, embedded in an HTML page and executed from within a browser
  - Unlike JavaScript, the Java code must be pre-compiled into a so-called *bytecode* before it can be interpreted by a browser's so–called Java Virtual Machine
  - In other words, the Notepad and the browser alone are not enough to write java applets

Client Side Technologies

- ActiveX controls
  - Similar to Java Applets but can be written in a variety of programming languages such as C, C++, VB and even Java
  - Supported by Windows only
  - Security issues: unlike Java applets, ActiveX controls have full access to all desktop resources: memory, operating systems, …
    - Authentication and registration system

Client Side Technologies

- Macromedia Flash
  - Proprietary commercial application for creating interactive graphic content
  - It has its own scripting language
  - To reproduce the Flash content browsers must be equipped with a Flash Player plug-in

Server Side Technologies

Server-side technologies are quite numerous and diverse. Popular server side web application technologies include:

- Microsoft ASP/.NET
- Java server technologies such as J2EE, JSP, and servlets
- CGI / Perl
- PHP
- ColdFusion

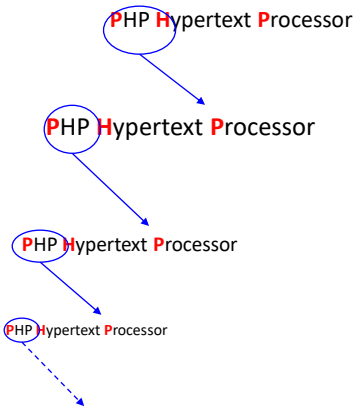
Server Side Technologies

- In addition, the server-side technologies include database systems such as Oracle, SQL Server (Microsoft), MySQL (open source) and many others
  - DB systems are indispensable part of server side operations and some DB software providers, such as Oracle are combining web application functionality with their core database functions

Server Side Technologies

- The “core” server side application development platforms can retrieve, modify and query the contents of databases through their own access mechanisms:
  - ADO.NET for Microsoft’s .NET platform enables access to almost every existing database platform
  - php enables direct access to many existing DB platforms, most notably MySQL, but also, Oracle, SQL Server and others

What does **PHP** stand for?



Recursive acronym!

Not Personal Home Page

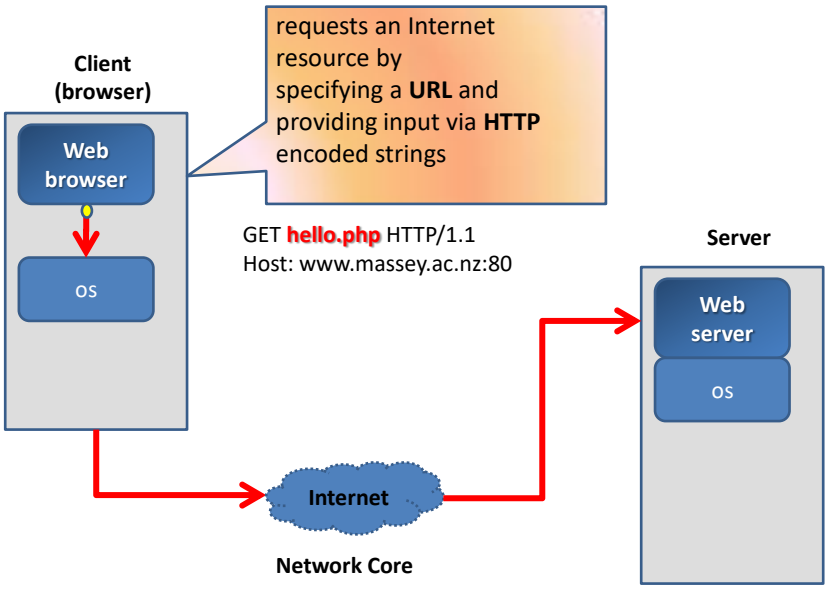
What is **PHP**?

- Server-side programming language
- Designed for quick development of HTML based dynamic web pages
  - –Server side scripts embedded in HTML pages
  - –**<?php** …… **?>**
- Elements of C, Java, and Perl

Evolution of PHP

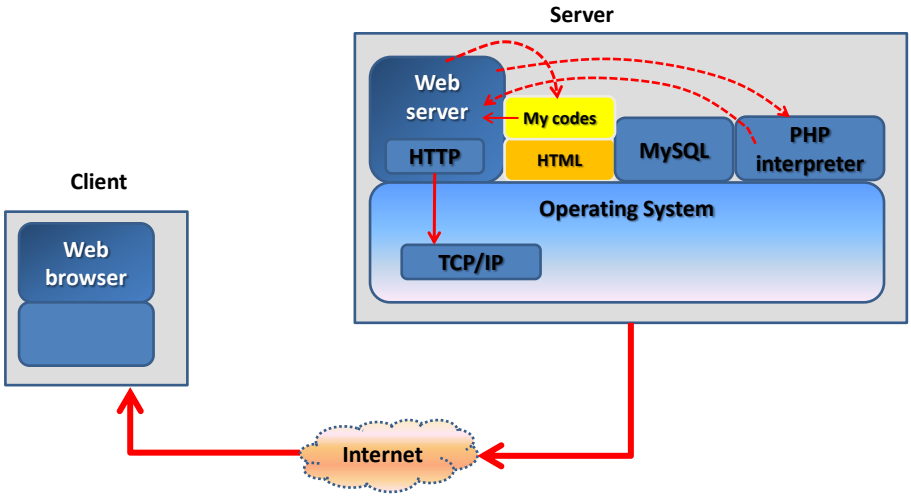
- 1994, *Rasmus Lerdorf*: started off as a series of Perl scripts
- June 8, 1995: Personal Home Page/Forms Interpreter, released to the public
- June, 1998, PHP3
- PHP4 introduced features of object-oriented programming
  - Objects/classes
  - Pass/return by reference
- PHP5 added more OO features
  - Exception handling
  - Abstract classes and methods

Client: makes a request

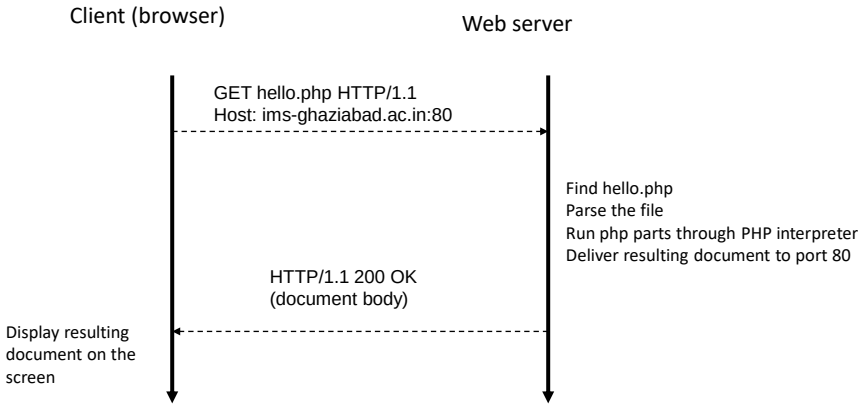


Server: responds

- **Webserver supports HTTP.**



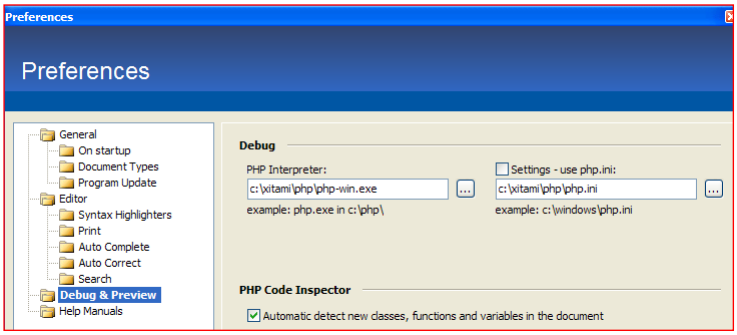
What's happening?



PHP Designer

Configure your PHP editor to access the PHP interpreter

From the menu bar, select **Tools/Preferences**



1st PHP program

Enter this text in a file called “hello.php”

```
<html>
<head><title>Hello</title></head>
<body>

<?php
   print "<h1>Hi there!</h1>";
?>

</body>
</html>
```

Load the URL in a browser and see what happens

Comparison with Javascript

Javascript is sent from server and runs on the client side.

PHP runs on the server, just like **CGI** scripts.
The client does not see the PHP code - **only the results**.

But PHP has some advantages.

Interpreted language

PHP is an interpreted language, i.e. an interpreter runs the code directly without compiling

Interpreted languages include: **PHP, Perl, Python**

Compiled languages include: **C, C++, Java**

PHP code is **executed** on the **server**, and the plain **HTML result** is sent to the **browser.**

Basic Syntax

• a PHP scripting block always starts with **<?php** and ends with **?>**.
• a PHP scripting block can be placed anywhere in the document.

• contains **HTML tags**, just like an HTML file, and some PHP scripting code.

• each code line in PHP must end with a **semicolon**.

• The file must have a **.php extension**. If the file has a **.html** extension, the PHP code will not be executed.

- possible extensions: "**.php**", "**.php3**", "**.php4**", "**.php5**", "**.phtml**",
- "**.inc**", "**.tpl**",

PHP Variables

Variables in PHP are **dynamically typed** - no need to declare them
- PHP is a weakly-typed programming language

Variable names begin with **$** then a character (**not number**)

✓ **$value** = 1;

✓ **$x** = 1.432344;

✓ **$myName** = "Rasmus Lerdorf";

✘ **yourName** = "Zeev Suraski"; **//invalid, missing $**

✘ **$7eleven** = "McGyver"; **//invalid, starts with a number**

PHP supports **references** (BE CAREFUL syntax is slightly different from C++!)
$a = **&**$x;

Assign by reference (an alias)

To assign by reference, simply place an ampersand (**&**) at the beginning of the variable which is being assigned (the source variable). For instance, the following code snippet outputs '*My name is Angus*' twice:

```
<?php

$foo = 'Angus';              // Assign the value 'Angus' to $foo
$bar = &$foo;                // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $bar;
echo $foo;                   // $foo is altered too.

?>
```

Try the following

Modify "hello.php" to print your name after assigning it to a variable.

What happens if this line is inserted into the php script?

```
print "<small>" . date("Y M d", getlastmod()) . "</small>";
```

What happens if you call **phpinfo()**?

phpinfo() is a valuable debugging tool as it contains all EGPCS (Environment, GET, POST, Cookie, Server) data.

Jumping in and out of PHP

```
<html><head><title>1st PHP</title></head><body>

    <?php
    print "This line was PHP generated";
    ?>
   <p> This line was not... </p>
    <?php
    print "This line was also PHP generated";
    ?>

</body></html>
```

Comments

C-style:
// this is a comment

Shell script style:
this is a comment

Multi-line comments:
/* this is a comment
this is the continuation of the comment */

Note: Nested multi-line comments are not allowed.

PHP Variables

Boolean	**boo**
Integer	**int**
Float	**flo**
String	**str**
Array	**arr**
Object	**obj**

PHP Variables

Convention:

```
$strName1='some name';
$intNumber1=2000000000;
$floNumber2=2.456;
$floNumber3=1.8e308; //notice scientific notation
```

PHP Constants

By default, case-sensitive as are variables

```
define("DEFAULT_SCRIPT", "php");
define("MIN_SPEED", 2);
```

```
define("DEFAULT_SCRIPT", "php",TRUE);
```

You can turn a constant variable case-insensitve using a third argument set to TRUE.

PHP Operators

=, ==, +, -, /, *, &&, ||, !, ++, --, %, /=, *=, >, <, >=, <=, &, |, ^, ~

– **All similar to C**

Additionally,

.	**String concatenation** $fullName = $firstName . " " . $lastName;
===, !===	**Identical test operator**: **same type** as well **as same value**
@	**Error suppression command** When placed in front of an expression in PHP, any error messages that might be generated by that expression will be ignored.
``	**Back tick** to execute **shell commands** (be careful for variations on different platforms)

Execution Operator (` `)

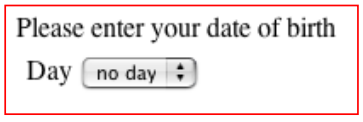
Note that these are not single-quotes!

PHP will attempt to execute the contents of the **backticks** as a shell command;

```
<?php
  $output = `dir *.php`;
  echo "<pre>$output</pre>";
?>
```

HTML with Form

Consider this html source to produce a simple form.



Very cumbersome and error prone!

```
<html>
<head><title>Date of Birth</title></head>
<body>
<p>Please enter your date of birth </p>
<form action="dataEntry.php" method="post">
 <table cellspacing="5">
   <tr><td valign="middle" colspan=2> Day
   <!-- drop-down list for days -->
   <select name=day>
                    <option selected="selected" value=""> no day </option>
                     <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                    <option value="4">4</option>
                    <option value="5">5</option>
                    <option value="6">6</option>
                    <option value="7">7</option>
                    <option value="8">8</option>
                    <option value="9">9</option>
                    <option value="10">10</option>
                    <option value="11">11</option>
                    <option value="12">12</option>
                    <option value="13">13</option>
                    <option value="14">14</option>
                    <option value="15">15</option>
                    <option value="16">16</option>
                    <option value="17">17</option>
                    <option value="18">18</option>
                    <option value="19">19</option>
                    <option value="20">20</option>
                    <option value="21">21</option>
                    <option value="22">22</option>
                    <option value="23">23</option>
                    <option value="24">24</option>
                    <option value="25">25</option>
                    <option value="26">26</option>
                    <option value="27">27</option>
                    <option value="28">28</option>
                    <option value="29">29</option>
                    <option value="30">30</option>
                    <option value="31">31</option>
     </select>
   </td></tr>
 </table>
</form> </body> </html>
```

PHP makes it easier

```
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23">23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>
```

```
<?php
 for ($day = 1; $day<= 31; ++$day) {
    print "\t<option value=\"$day\">$day</option>\n";
 }
?>
```

Note: use the escape sequence **\"** to produce double quotes

Escape Characters

\n	line feed character
\r	carriage return character
\t	tab
****	backslash character
\$	dollar sign
\"	double quote character

Arrays

An array in PHP is actually an ordered map.
A map is a type that associates ***values*** to ***keys***.

```
array( key => value , ... )

// key may only be an integer or string
// value may be any value of any type
```

Escape Characters

\n	line feed character
\r	carriage return character
\t	tab
****	backslash character
\$	dollar sign
\”	double quote character

Arrays

An array in PHP is actually an ordered map.
A map is a type that associates ***values*** to ***keys***.

```
array( key => value , ... )

// key may only be an integer or string
// value may be any value of any type
```

Creating & printing Arrays

```
// Create a simple array.
$array = array(1, 2, 3, 4, 5);
print_r($array);
```

Output:

```
Array (
[0] => 1
[1] => 2
[2] => 3
[3] => 4
[4] => 5
)
```

Printing Array Elements

```
<?php
 $array = array(5=>43, 32, 56, "b"=> 12);
 print("\$array[\"b\"]="); ✓
 echo $array["b"]; ✓
 print "<br>"; ✓
 print("\$array[\"b\"]= $array['b']"); ×
 print("\$array[\"b\"]= {$array['b']}"); ✓
 print("\$array[\"b\"]= {$array[“b”]}"); ✓
 print("\$array[\"b\"]= $array["b"]"); ×
?>
```

Output of the correct statements: $array["b"]=12

More Print Examples

```
<?php
      print "<br>";
      print("Hello World<br>");
      print "Hello World<br>";  //print() also works without parentheses
      $number=7;
      print $number;  //you can just print variables without double quotes
      print '<br>$number';  //this will print the variable name.
?>
```

Output:

Hello World
Hello World
7
$number

Extending & Modifying Arrays

```
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;    // This is the same as $arr[13] = 56;
                // at this point of the script

$arr["x"] = 42; // This adds a new element to
                // the array with key "x"

unset($arr[5]); // This removes the element from the array

unset($arr);    // This deletes the whole array
?>
```

int array_unshift()

Prepend one or more elements to the beginning of an array

Note that the list of elements is prepended as a whole, so that the prepended elements stay in the same order. All numerical array keys will be modified to start counting from zero while literal keys won't be touched.
-returns the **new** number of elements.

```
<?php
$queue = array("orange", "banana");
array_unshift($queue, "apple", "raspberry");
print_r($queue);
?>
```

```
Array (
 [0] => apple
 [1] => raspberry
 [2] => orange
 [3] => banana
)
```

int array_push()

Push one or more elements onto the end of array.
Returns the new number of elements

```
<?php
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
    [3] => raspberry
)
```

mixed array_pop()

pops and returns the last value of the array, shortening the array by one element. If array is empty (or is not an array), NULL will be returned. Will additionally produce a Warning when called on a non-array.

```
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_pop($stack);
print_r($stack);
?>
```

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
)
```

PHP arrays: more examples

Arrays in PHP are associative arrays, i.e. indices need not be integers.

```
$details = array("name"=>"Angus McGyver", "dept"=>"IIMS",
          "city=>"Auckland");

print $details["name"];
```

Features auto-indexing, e.g.:

```
$daysOfWeek = array(1=>"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun");
$myNumbers = arrray(1, 2, 3, 4, 5, 10=>1, 5=>2, 20, 3=>25);
```

Iterating through Arrays

```
foreach (array_expression as $value) {
    statement
}
```

Example:

```
<html>
<body>
<?php

$x=array("red","green","blue");
foreach ($x as $value) {
    echo $value . "<br />";
}

?>

</body>
</html>
```

- Loops over the array given by *array_expression*.
- On each loop, the value of the current element is assigned to *$value* and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).
- Note: acts on a copy of the array

Output:

```
red
green
blue
```

Iterating through Arrays

foreach (array_expression as **$key** => **$value**)
  statement

Example:

```
<html>
<body>
<?php

$numbers = array("one"=>"une", "two"=>"deux",
"three"=>"trois", "four"=>"quatre", "five"=>"cinq");

foreach($numbers as $key=>$val) {
  print "English: " . $key . ", French " . $val . "<br/>";
}

?>

</body>
</html>
```

- Loops over the array given by *array_expression*.
- On each loop, the current element's key is assigned to the variable *$key*.
- Note: acts on a copy of the array

Output:

```
English: one, French une
English: two, French deux
English: three, French trois
English: four, French quatre
English: five, French cinq
```

Try this!

Type in the arrays in a test program and try
print_r($arrayName);

Modify the date of birth PHP script so that the drop-down list for the months displays the month name ("January", "February", etc.) rather than the month number.

PHP Control Structures

- if, else, switch … case …, break, continue
  - Similar to C, but must use brackets {} always
- elseif
  - Different syntax to C, same semantics
- While loop …, do … while loop, for loop…
  - All similar to C

PHP Control Structures

```
switch(expression){
  case $Example:  //variable name
            statements;
            break;
  case “text”:
              //…
  case 75:
             //…
  case -123.45:
             //…
   default:
             //…
}
```