Web Development for Web Developers

Administrative Order No. 39

Administrative Order No. 39

MANDATING GOVERNMENT AGENCIES TO MIGRATE TO THE GOVERNMENT WEB HOSTING SERVICE (GWHS) OF THE DEPARTMENT OF SCIENCE AND TECHNOLOGY-INFORMATION AND COMMUNICATIONS TECHNOLOGY OFFICE (DOST-ICTO)

Administrative Order No. 39

Section 24, Article II of the 1987 Constitution provides that the State shall recognize the vital role of communication and information in nation-building;

Section 2(b) of Executive Order (EO) No. 47 (s. 2011) mandates the ICTO, under the DOST, to provide an efficient information and communications technology infrastructure, information systems and resources to support an effective, transparent and accountable governance and, in particular, support the speedy enforcement of rules and delivery of accessible public services to the people; and

the government recognizes the need for greater security and robustness in the internet technologies it uses to deliver reliable information and provide state of the art online services, which are free from impairment and disruption.

Administrative Order No. 39

Section 1

Establishment of the Government Web Hosting Service (GWHS). DOST-ICTO shall administer, operate and maintain the GWHS to ensure the government's internet presence 24-hours a day, 7-days a week (24/7) under all foreseeable conditions.

Administrative Order No. 39

Section 2

Responsibilities of the Implementing Agencies. DOST-ICTO shall administer, operate and maintain the GWHS, pursuant to the policies, rules and regulations set by the National Digital Service Coordinating Group (NDSCG), and shall ensure the efficiency, integrity and reliability of the GWHS, the websites and online services that the GWHS hosts.

Content Coverage. NDSCG shall be responsible for issuance of the Philippine Uniform Website Content Policy (UWCP) and its implementing rules and regulations. On the other hand, the Presidential Communications Development and Strategic Planning Office (PCDSPO), shall administer the implementation of the UWCP under the direct control and supervision of the NDSCG.

Technical Assistance. DOST-ICTO shall provide the technical assistance the PCDSPO shall need to develop and implement the policies, rules and regulations that NDSCG will prescribe.

Administrative Order No. 39

Section 3

Responsibilities of Other Government Agencies. The responsibilities of hosted agencies are as follows:

a) Produce and regularly update the contents of their websites, subject to the UWCP and its implementing rules;

b) Develop online services offered through their websites, subject to prevailing policies, rules and regulations of the government; and

c) Employ a webmaster duly designated by the Head of Agency.

Administrative Order No. 39

Section 4

Migration of Existing Website. All NGAs, GFIs, GOCCs, and inter-agency collaborations, programs and projects shall completely migrate their websites to the GWHS, without prejudice to contractual rights of the existing web hosting providers, if any, within one (1) year from the effectivity of this AO.

The Department of the Interior and Local Government (DILG), with the assistance of the DOST-ICTO, shall advocate to LGUs the adoption of UWCP and the migration of their websites to the GWHS.

Administrative Order No. 39

SECTION 5

Government IP Exchange and Data Peering. The DOST-ICTO shall establish a Government Internet Protocol (IP) Exchange (G/IPX) facility. All government agencies shall ensure that they exchange data traffic with other government agencies and external stakeholders through this G/IPX facility.

Administrative Order No. 39

SECTION 6

Cost of Operating the Facilities and Services of the GWHS. The DOST-ICTO shall include in its annual appropriations the amounts necessary for the personnel services and its maintenance and other operating expenses. The DOST-ICTO may be allowed to charge fees from its subscribers for the use of its facilities and services on a cost recovery basis to fund its variable expenses, in accordance with the provisions of AO No 31 (s. 2012).

Administrative Order No. 39

SECTION 7

SECTION 7. Use of the Electronic Government Fund (e-Gov Fund).

The DOST-ICTO is hereby authorized to set aside a Website Migration Fund to be sourced from e-Gov Fund, subject to existing laws, rules and regulation.

Administrative Order No. 39

SECTION 8

Reports.

The NDSCG shall submit reports, annually and/or as often as maybe required by the President, relative to the implementation of the provisions of this AO.

Administrative Order No. 39

SECTION 9

Repealing Clause.

All issuances, orders, rules and regulations or parts thereof which are inconsistent with the provisions of this AO are hereby repealed, amended or modified accordingly.

Administrative Order No. 39

SECTION 10

Separability Clause.

Should any provision of this AO be declared invalid or unconstitutional, the other provisions not affected thereby shall remain valid and subsisting.

Administrative Order No. 39

SECTION 11

Effectivity.

This AO shall take effect immediately.

Open Forum

Web Development for Web Developers

Module 1: Basic Web Development
- HTML
- CSS
- Bootstrap
- Javascript

Module 2: Advanced Web Development
- PHP
- SQL Database
- Deployment

Module 1: Basic Web Development

HTML

Hyper**T**ext **M**arkup **L**anguage

- defines the structure of your content
- consists of a series of elements, which you use to enclose, or wrap, different parts of the content

Structure of an HTML file:

The opening tag:

The closing tag:

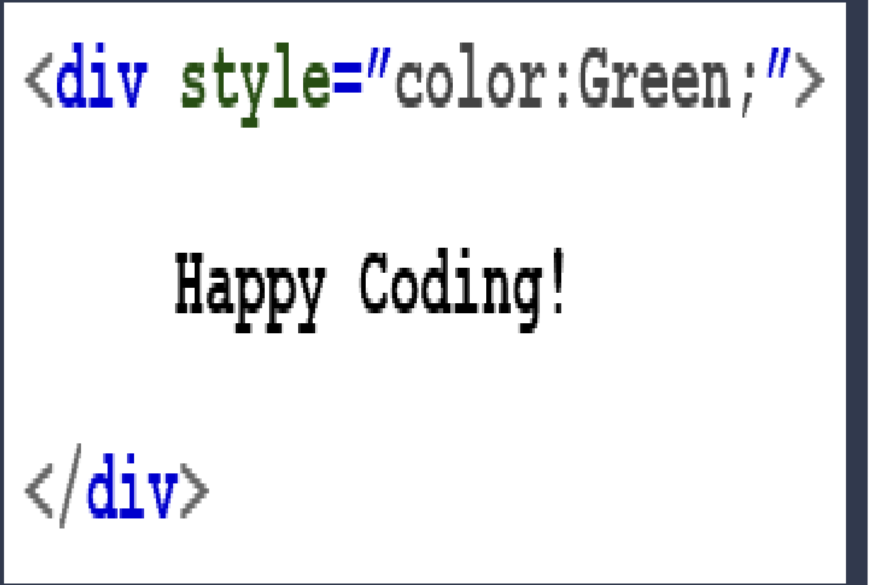
The content:

The element:

Attributes:

Analyze the structure of the html file

```
<!doctype html>

<html>

          <head>

                    <meta charset="utf-8"
/>

          <meta name="viewport"
          content="width=device-width" />

                    <title>

          Web Development for Web
          Developers

          </title>

          </head>

          <body>

          <div style=”color:Green;”>

               Happy Coding!

          </div>

     </boby>

</html>
```



```
<div style="color:Green;">

    Happy Coding!

</div>
```

<title> - opening tag

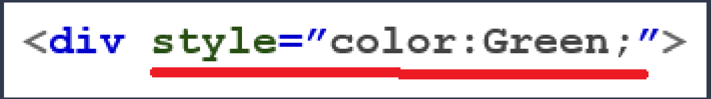
Sample HTML - Content

</title> - closing tag

<title>Sample HTML</title>

Element

Setting attributes a value



```
<div style=″color:Green;″>
```

Note: Simple attribute values that don't contain ASCII whitespace (or any of the characters " ' ` = < >) can remain unquoted, but it is recommended that you quote all attribute values, as it makes the code more consistent and understandable.

Attributes that set a value always have:

- A space between it and the element name (or the previous attribute, if the element already has one or more attributes).
- The attribute name followed by an equal sign.
- The attribute value wrapped by opening and closing quotation marks.

Nesting elements

You can put elements inside other elements too — this is called nesting.

Attributes applied to the lowest level element will prevail over the attributes from its parent.

make sure that your elements are properly nested.

```
<div style="color:green;">

       <div
style="color:red;">

              <p>

              What's my Color?

              </p>

       </div>

</div>
```

Nesting elements

You can put elements inside other elements too — this is called nesting.

Attributes applied to the lowest level element will prevail over the attributes from its parent.

```
<div style="color:green;">
        <div style="color:red;">
                <p>
                What's my Color?
                </p>
        </div>
        <div>
        How about me, what’s my color?
        </div>
</div>
```

Note on Nesting Elements

The elements have to open and close correctly so that they are clearly inside or outside one another. If they overlap as shown above, then your web browser will try to make the best guess at what you were trying to say, which can lead to unexpected results.

Void elements

Some elements have no content and are called **void elements**.

```
<img src="images/firefox-icon.png"
alt="My test image" />
```

This contains two attributes, but there is no closing </img> tag and no inner content. This is because an image element doesn't wrap content to affect it. Its purpose is to embed an image in the HTML page in the place it appears.

- `base`
- `br`
- `col`
- `embed`
- `hr`
- `img`
- `input`
- `keygen`
- `link`
- `meta`
- `param`
- `source`
- `track`
- `wbr`

Anatomy of an HTML document

individual elements are combined to form an entire HTML page.

```
<!doctype html>

<html>

          <head>

                    <meta charset="utf-8"
/>

          <meta name="viewport"
          content="width=device-width" />

                    <title>

          Web Development for Web
          Developers

          </title>

          </head>

          <body>

          <div style=”color:Green;”>

               Happy Coding!

          </div>

     </boby>

</html>
```

Anatomy of an HTML document

<!DOCTYPE html>

Doctype -

It is a required preamble. Old days uses this to make error trapping easy inside the html structure. It allows the browser behaves in accordance with html rendering.

Anatomy of an HTML document

<html></html>

the <html> element.

This element wraps all the content on the entire page and is sometimes known as the root element. It also includes the lang attribute, setting the primary language of the document.

Anatomy of an HTML document

<head></head>

the <head> element.

This element acts as a container for all the stuff you want to include on the HTML page that isn't the content you are showing to your page's viewers. This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.

Anatomy of an HTML document

```
<meta charset="utf-8">
```

This element sets the character set your document should use to UTF-8 which includes most characters from the vast majority of written languages. Essentially, it can now handle any textual content you might put on it. There is no reason not to set this, and it can help avoid some problems later on.

Anatomy of an HTML document

```
<meta
name="viewport"
content="width=device-
width">
```

This viewport element ensures the page renders at the width of viewport, preventing mobile browsers from rendering pages wider than the viewport and then shrinking them down.

Anatomy of an HTML document

<title></title>

the <title> element.

This sets the title of your page, which is the title that appears in the browser tab the page is loaded in. It is also used to describe the page when you bookmark/favorite it.

Anatomy of an HTML document

<body></body>

the <body> element.

This contains all the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.

HTML text fundamentals

Learn how to mark up a basic page of text to give it structure and meaning — including paragraphs, headings, lists, emphasis, and quotations.

One of HTML's main jobs is to give text structure so that a browser can display an HTML document the way its developer intends.

The basics: headings and paragraphs

Most structured text consists of headings and paragraphs, whether you are reading a story, a newspaper, a college textbook, a magazine, etc.

HTML text fundamentals

Structured content makes the reading experience easier and more enjoyable.

Paragraph

```
<p>I am a paragraph, oh yes I am.</p>
```

heading elements

```
<h1>Heading 1</h1>

<h2>Heading 1</h2>

<h3>Heading 1</h3>

<h4>Heading 1</h4>

<h5>Heading 1</h5>

<h6>Heading 1</h6>
```

HTML text fundamentals

heading elements

There are six heading elements: h1, h2, h3, h4, h5, and h6. Each element represents a different level of content in the document; <h1> represents the main heading, <h2> represents subheadings, <h3> represents sub-subheadings, and so on.

HTML text fundamentals

Lists

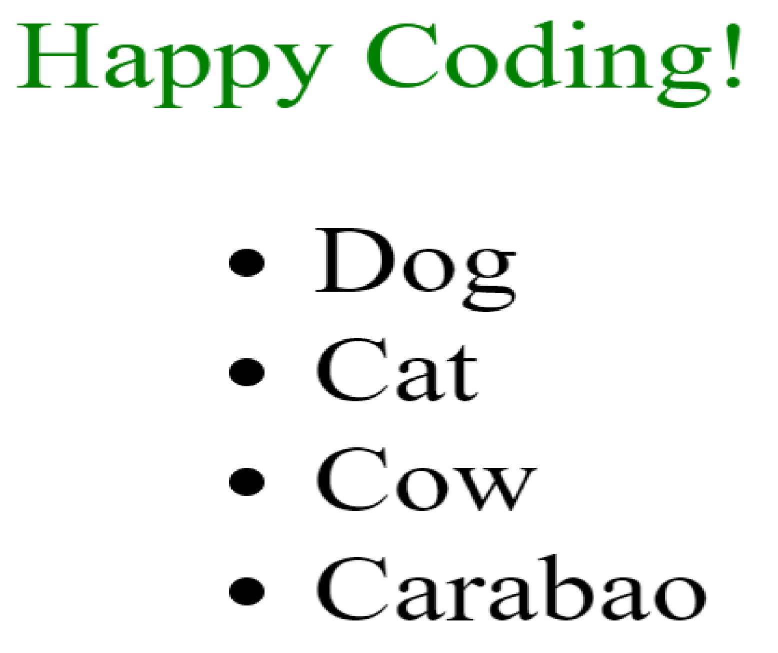
Unordered

Unordered lists are used to mark up lists of items for which the order of the items doesn't matter.

Every unordered list starts off with a <ul> element

The last step is to wrap each list item in a <li> (list item) element

```
<ul>

  <li>Dog</li>

  <li>Cat</li>

  <li>Cow</li>

  <li>Carabao</li>

</ul>
```



Happy Coding!

- Dog
- Cat
- Cow
- Carabao

```
     <body>
     <div style=”color:Green;”>
          <h1>Happy Coding!<h1>
     </div>
     <div>
          <ul>

     <li>Dog</li>

     <li>Cat</li>

     <li>Cow</li>

     <li>Carabao</li>
          </ul>
     </div>
</boby>
```

HTML text fundamentals

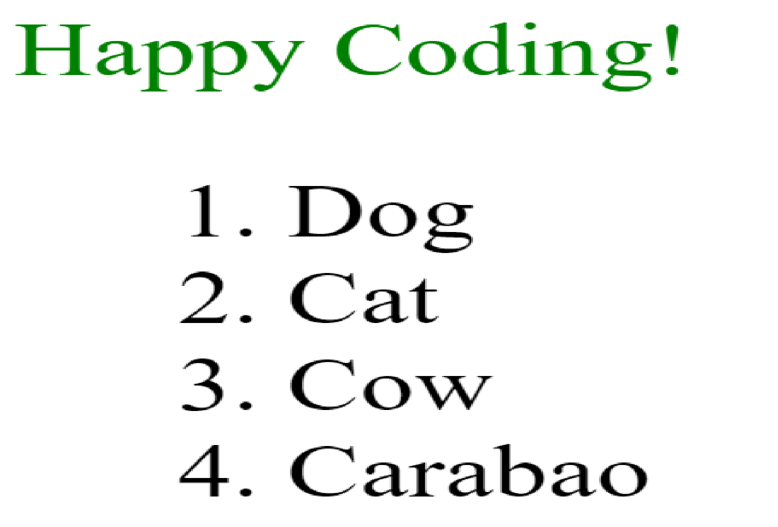
Lists

Ordered

Ordered lists are lists in which the order of the items does matter.

The markup structure is the same as for unordered lists, except that you have to wrap the list items in an <ol> element, rather than <ul>:

```
     <ol>

  <li>Dog</li>

  <li>Cat</li>

  <li>Cow</li>

  <li>Carabao</li>

     </ol>
```



Happy Coding!

1. Dog
2. Cat
3. Cow
4. Carabao

```
    <body>

    <div style=”color:Green;”>

        Happy Coding!

    </div>

    <div>

        <ol>


    <li>Dog</li>


    <li>Cat</li>


    <li>Cow</li>


    <li>Carabao</li>

        </ol>

    </div>

</boby>
```

The Form element

The <form> HTML element represents a document section containing interactive controls for submitting information.

```
<form action="" method="get">
  <div>
              <label for="name">Enter your name: </label>
              <input type="text" name="name" id="name"
required />
  </div>
  <div>
              <label for="email">Enter your email:
</label>
              <input type="email" name="email" id="email"
required />
  </div>
  <div>
              <input type="submit" value="Subscribe!" />
  </div>
</form>
```

The Form element

Attributes for form submission

action

The URL that processes the form submission. This value can be overridden by a formaction attribute on a <button>, <input type="submit">, or <input type="image"> element. This attribute is ignored when method="dialog" is set.

method

The HTTP method to submit the form with. The only allowed methods/values are (case insensitive):

- post: The POST method; form data sent as the request body.
- get (default): The GET; form data appended to the action URL with a ? separator. Use this method when the form has no side effects.

The Form element

The <input> HTML element is used to create interactive controls for web-based forms in order to accept data from the user; The <input> element is one of the most powerful and complex in all of HTML due to the sheer number of combinations of input types and attributes.

```
<label for="name">Name (4 to 8
characters):</label>

<input type="text" id="name" name="name"
required minlength="4" maxlength="8"
size="10" />
```

Other input types

button	date	file	month
checkbox	datetime	hidden	number
color	email	image	password
radio	range	search	submit

Open Forum

Web Development for Web Developers

CSS - Cascading Style Sheet

Module 1: Basic Web Development

CSS

Cascading Style Sheet

- CSS (Cascading Style Sheets) is the code that styles web content

- Like HTML, CSS is not a programming language. It's not a markup language either. CSS is a style sheet language. CSS is what you use to selectively style HTML elements.

Reference CSS on HTML file

External CSS

Inside the HTML header insert the code below:

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

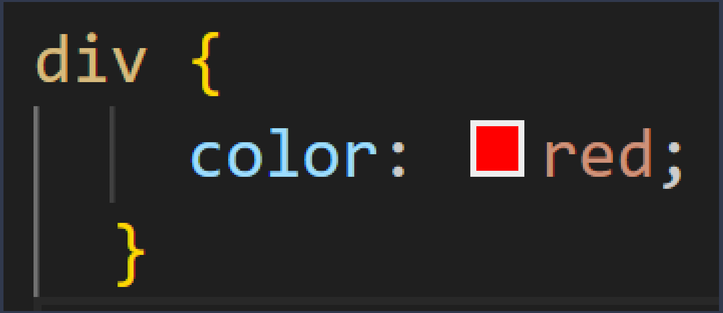
Internal CSS

```
<style>

</style>
```

Inline CSS

```
<h1 style="color:blue;">A Blue Heading</h1>
```

Anatomy of a CSS ruleset



```
div {
    color: red;
  }
```

div - Selector

Color: red; - declaration

Color: - property

Red - value

Anatomy of a CSS ruleset

The whole structure is called a ruleset. (The term ruleset is often referred to as just rule.) Note the names of the individual parts:

Selector

This is the HTML element name at the start of the ruleset. It defines the element(s) to be styled. To style a different element, change the selector.

Declaration

This is a single rule like color: red;. It specifies which of the element's properties you want to style.

Anatomy of a CSS ruleset

Properties

These are ways in which you can style an HTML element.

Property value

To the right of the property—after the colon—there is the property value. This chooses one out of many possible appearances for a given property.

CSS Syntax explain

- Apart from the selector, each ruleset must be wrapped in curly braces. ({})

- Within each declaration, you must use a colon (:) to separate the property from its value or values.

- Within each ruleset, you must use a semicolon (;) to separate each declaration from the next one.

multiple property values in one ruleset are separated by semicolons,

```
div {
        color: red;
        width: 500px;
        border: 1px solid
black;
   }
```

Selecting multiple elements

Separate multiple selectors by commas

```
p, li, h1 {
  color: red;
}
```

Different types of selectors

Element selector

All HTML elements of the specified type.

P, div, input etc.

```
div {
        color: red;
        width: 500px;
        border: 1px solid
black;
    }
```

Different types of selectors

ID selector

The element on the page with the specified ID. On a given HTML page, each id value should be unique.

```
#my_div {
        color: green;
        width: 500px;
        border: 1px solid
black;
   }
```

Different types of selectors

Class selector

The element(s) on the page with the specified class. Multiple instances of the same class can appear on a page.

```
#my_first_class {

        color: blue;

        width: 500px;

        border: 1px solid
black;

   }
```

Different types of selectors

Attribute selector

The element(s) on the page with the specified attribute.

```
#my_first_class {
        color: blue;
        width: 500px;
        border: 1px solid
black;
   }
```

CSS Colors

RGB and RGBA

In HTML, a color can be specified as an RGB value, using this formula

Each parameter (red, green, and blue) defines the intensity of the color with a value between 0 and 255.

rgb(red, green, blue)

rgb(255, 0, 0)
rgb(0, 0, 255)
rgb(0, 255, 0)
rgb(106, 90, 205)

CSS Colors

HEX Color Values

In HTML, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

#ff0000

#0000ff

#00ff00

#6a5acd

CSS Colors

HSL Color Values

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(hue, saturation, lightness)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value. 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage value. 0% is black, and 100% is white.

hsl(0, 100%, 50%)
hsl(240, 100%, 50%)
hsl(130, 100%, 45%)
hsl(300, 76%, 72%)

CSS backgrounds

CSS background image

```
body {

  background-image: url("img_tree.gif");

}
```

CSS background repeat

```
body {

  background-image: url("img_tree.gif");

  background-repeat: no-repeat, repeat;

}
```

CSS backgrounds Attachments

scroll	The background image will scroll with the page. This is default
fixed	The background image will not scroll with the page
local	The background image will scroll with the element's contents
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element

CSS background Attachment

```
body {

  background-image: url("img_tree.gif");

  background-repeat: no-repeat;

   background-attachment: fixed;

}
```

A background-image that will not scroll with the page (fixed)

CSS background shorthand

To shorten the code, it is also possible to specify all the background properties in one single property.

```
body {

  background: #ffffff url("img_tree.png")
no-repeat right top;

}
```

Shorthand Order

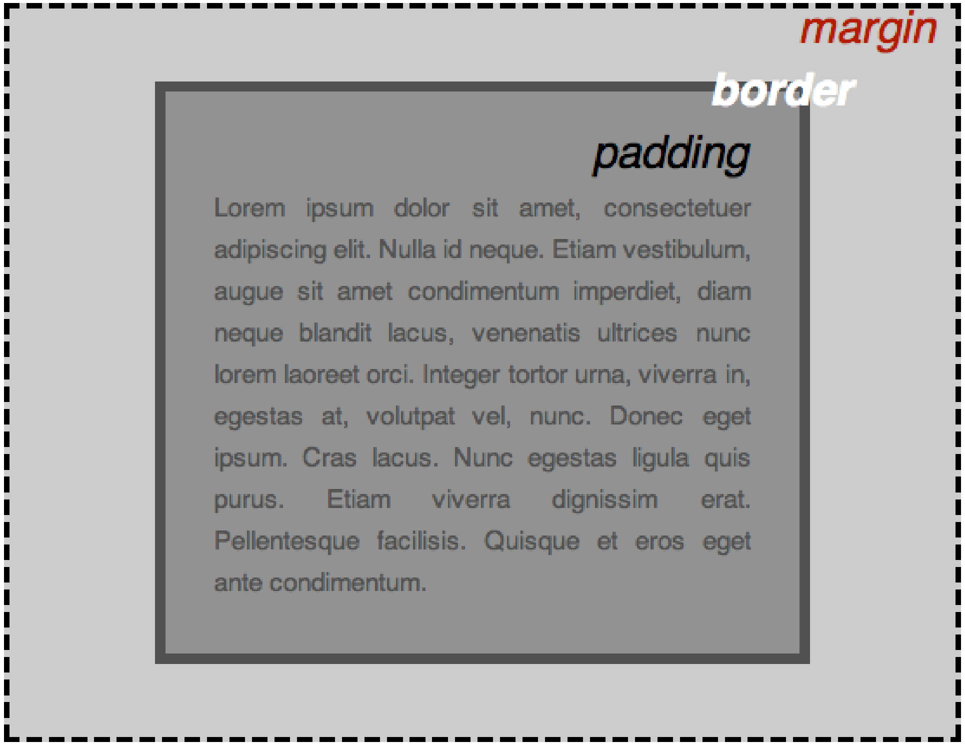
```
background-color

background-image

background-repeat

background-attachment

background-position
```

CSS in a Box

CSS layout is mostly based on the box model. Each box taking up space on your page has properties

- padding, the space around the content. In the example below, it is the space around the paragraph text.
- border, the solid line that is just outside the padding.
- margin, the space around the outside of the border.

Visualizing the margin, border and padding.



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla id neque. Etiam vestibulum, augue sit amet condimentum imperdiet, diam neque blandit lacus, venenatis ultrices nunc lorem laoreet orci. Integer tortor urna, viverra in, egestas at, volutpat vel, nunc. Donec eget ipsum. Cras lacus. Nunc egestas ligula quis purus. Etiam viverra dignissim erat. Pellentesque facilisis. Quisque et eros eget ante condimentum.

Some other styling used to achieve the desired effect

- width (of an element).
- background-color, the color behind an element's content and padding.
- color, the color of an element's content (usually text).
- text-shadow sets a drop shadow on the text inside an element.
- display sets the display mode of an element. (keep reading to learn more).

More CSS Rules

Html page and body styling

Changing the page color:

```
html {
  background-color: #00539f;
}
```

Styling the body

```
body {
  width: 600px;
  margin: 0 auto;
  background-color: #ff9500;
  padding: 0 20px 20px 20px;
  border: 5px solid black;
}
```

More CSS Rules

Positioning and styling the main page title

```
h1 {

  margin: 0;

  padding: 20px 0;

  color: #00539f;

  text-shadow: 3px 3px 1px black;

}
```

More CSS Rules

Image

```
img {
  display: block;
  margin: 0 auto;
}
```

Open Forum

Web Development for Web Developers

Bootstrap

Bootstrap

Bootstrap is a powerful, feature-packed frontend toolkit. Build anything—from prototype to production—in minutes.

Include Bootstrap's CSS and JS.

Place the <link> tag in the <head> for our CSS, and the <script> tag for our JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing </body>. Learn more about our CDN links.

Referencing Bootstrap to HTML File

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="/styles/bootstrap.min.css" rel="stylesheet" />
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="/scripts/bootstrap.bundle.min.js"></script>
  </body>
</html>
```

Important globals

Bootstrap employs a handful of important global styles and settings, all of which are almost exclusively geared towards the normalization of cross browser styles. Let’s dive in.

HTML5 doctype

Viewport meta

Box-sizing

Reboot

Important globals

HTML5 doctype

Bootstrap requires the use of the HTML5 doctype. Without it, you’ll see some funky and incomplete styling.

```
<!doctype html>

<html lang="en">

  ...

</html>
```

Important globals

Viewport meta

Bootstrap is developed mobile first, a strategy in which we optimize code for mobile devices first and then scale up components as necessary using CSS media queries. To ensure proper rendering and touch zooming for all devices, add the responsive viewport meta tag to your <head>.

```
<meta name="viewport"
content="width=device-
width, initial-scale=1">
```

Important globals

Box-sizing

For more straightforward sizing in CSS, we switch the global box-sizing value from content-box to border-box. This ensures padding does not affect the final computed width of an element, but it can cause problems with some third-party software like Google Maps and Google Custom Search Engine.

On the rare occasion you need to override it, use something like the following:

```
.selector-for-some-widget {

  box-sizing: content-box;

}
```

Important globals

Reboot

For improved cross-browser rendering, we use Reboot to correct inconsistencies across browsers and devices while providing slightly more opinionated resets to common HTML elements.

Compatible browsers

WEB

Chrome >= 60

Firefox >= 60

Firefox ESR

iOS >= 12

Safari >= 12

not Explorer <= 11

Compatible browsers

WEB

	MAC	WINDOWS
Chrome	Yes	Yes
Firefox	Yes	Yes
Microsoft Edge	Yes	Yes
Opera	Yes	Yes
Safari	Yes	

Compatible browsers

MOBILE

	iOS	Android
Chrome	Yes	Yes
Firefox	Yes	Yes
Safari	Yes	
Android Browser		V6.0

Containers

Containers are the most basic layout element in Bootstrap and are required when using our default grid system. Containers are used to contain, pad, and (sometimes) center the content within them. While containers can be nested, most layouts do not require a nested container.

Bootstrap comes with three different containers:

- .container, which sets a max-width at each responsive breakpoint
- .container-{breakpoint}, which is width: 100% until the specified breakpoint
- .container-fluid, which is width: 100% at all breakpoints

Container Behavior

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px	XX-Large ≥1400px
`.container`	100%	540px	720px	960px	1140px	1320px
`.container-sm`	100%	540px	720px	960px	1140px	1320px
`.container-md`	100%	100%	720px	960px	1140px	1320px
`.container-lg`	100%	100%	100%	960px	1140px	1320px
`.container-xl`	100%	100%	100%	100%	1140px	1320px
`.container-xxl`	100%	100%	100%	100%	100%	1320px
`.container-fluid`	100%	100%	100%	100%	100%	100%

Grids

Bootstrap’s grid system uses a series of containers, rows, and columns to layout and align content. It’s built with flexbox and is fully responsive.

```
<div class="container text-center">
  <div class="row">
           <div class="col">
           Column
           </div>
           <div class="col">
           Column
           </div>
           <div class="col">
           Column
           </div>
  </div>
</div>
```

Grids

Grid options

- Extra small (xs)
- Small (sm)
- Medium (md)
- Large (lg)
- Extra large (xl)
- Extra extra large (xxl)

Grid Properties

	xs <576px	sm ≥576px	md ≥768px	lg ≥992px	xl ≥1200px	xxl ≥1400px
Container `max-width`	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`	`.col-xxl-`
# of columns	12					

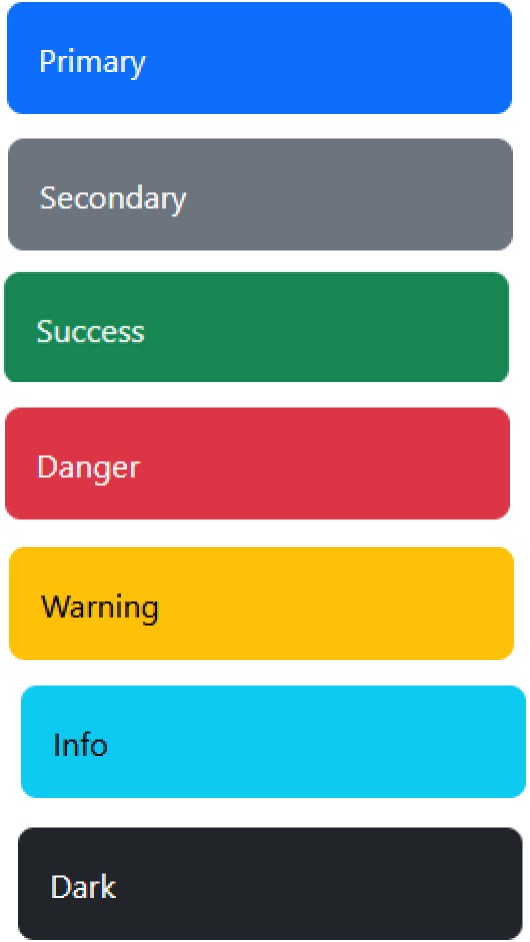
Text/Typography

Text alignment

```
<p class="text-start">Start aligned text on all
viewport sizes.</p>

<p class="text-center">Center aligned text on
all viewport sizes.</p>

<p class="text-end">End aligned text on all
viewport sizes.</p>

<p class="text-sm-end">End aligned text on
viewports sized SM (small) or wider.</p>

<p class="text-md-end">End aligned text on
viewports sized MD (medium) or wider.</p>

<p class="text-lg-end">End aligned text on
viewports sized LG (large) or wider.</p>

<p class="text-xl-end">End aligned text on
viewports sized XL (extra large) or wider.</p>

<p class="text-xxl-end">End aligned text on
viewports sized XXL (extra extra large) or
wider.</p>
```

Colors

Theme colors



Primary

Secondary

Success

Danger

Warning

Info

Dark

Tables

```
<table class="table">
  <thead>
         <tr>
         <th scope="col">#</th>
         <th scope="col">First</th>
         <th scope="col">Last</th>
         <th scope="col">Handle</th>
         </tr>
  </thead>
  <tbody>
         <tr>
         <th scope="row">1</th>
         <td>Mark</td>
         <td>Otto</td>
         <td>@mdo</td>
         </tr>

  </tbody>
</table>
```

Tables with theme

Apply on Table

```
<table class="table-primary">...</table>
```

```
<table class="table-secondary">...</table>
```

Apply on Rows

```
<tr class="table-primary">...</tr>
```

```
<tr class="table-secondary">...</tr>
```

Apply on Cells

```
 <td class="table-primary">...</td>
```

```
  <td class="table-secondary">...</td>
```

Images

Responsive images

Images in Bootstrap are made responsive with .img-fluid. This applies max-width: 100%; and height: auto; to the image so that it scales with the parent width.

```
<img src="..." class="img-fluid" alt="...">
```

Images

Image thumbnails

you can use .img-thumbnail to give an image a rounded 1px border appearance.

```
<img src="..." class="img-thumbnail"
alt="...">
```

Images

Aligning images

Align images with the helper float classes or text alignment classes.

```
<img src="..." class="rounded float-start"
alt="...">

<img src="..." class="rounded float-end"
alt="...">
```

```
<div class="text-center">

  <img src="..." class="rounded" alt="...">

</div>
```

Jumbotron

No longer continue in Version 5

```
<div class="container-fluid py-5">
         <h1 class="display-5 fw-bold">Custom jumbotron</h1>
         <p class="col-md-8 fs-4">Using a series of utilities, you can create this jumbotron, just like the one in previous versions of Bootstrap. Check out the examples below for how you can remix and restyle it to your liking.</p>
         <button class="btn btn-primary btn-lg" type="button">Example button</button>
         </div>
```

Alerts

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

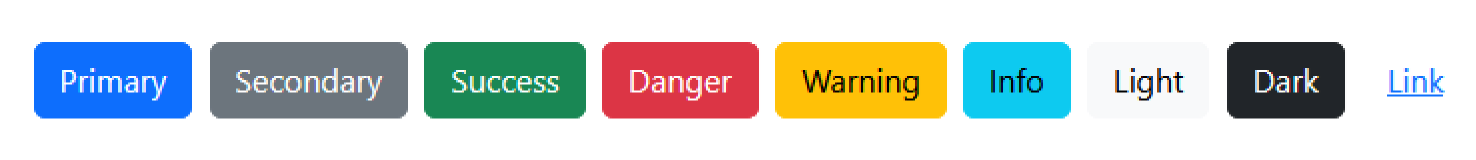
```
<div id="liveAlertPlaceholder"></div>

<button type="button" class="btn btn-primary"
id="liveAlertBtn">Show live alert</button>
```

It must be trigger by Javascript.

Buttons

Bootstrap has a base .btn class that sets up basic styles such as padding and content alignment.

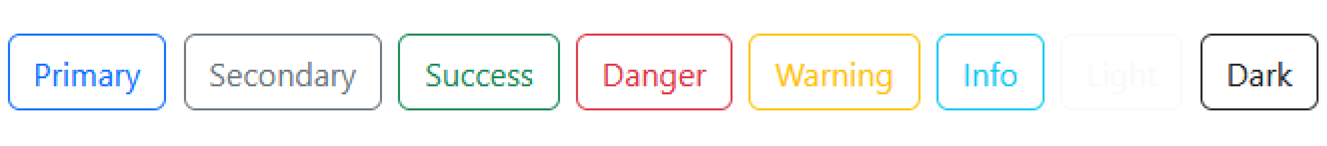
```
<button type="button" class="btn">Base class</button>

<button type="button" class="btn btn-primary">Primary</button>

<button type="button" class="btn btn-secondary">Secondary</button>

<button type="button" class="btn btn-success">Success</button>

<button type="button" class="btn btn-danger">Danger</button>
```



Outline buttons

Replace the default modifier classes with the .btn-outline-* ones to remove all background images and colors on any button.

```
<button type="button" class="btn btn-
outline-primary">Primary</button>

<button type="button" class="btn btn-
outline-secondary">Secondary</button>

<button type="button" class="btn btn-
outline-success">Success</button>

<button type="button" class="btn btn-
outline-danger">Danger</button>
```



Open Forum

Javascript

Javascript

- JavaScript is the world's most popular programming language.
- JavaScript is the programming language of the Web.
- JavaScript is easy to learn.
- This tutorial will teach you JavaScript from basic to advanced.

JavaScript Can Change HTML Content

Using Id
Using Name

```
document.getElementById("demo").innerHTML =
"Hello JavaScript";
```

```
 document.getElementsByName("fname").innerHTML
= "Hello JavaScript";
```

Variables

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

Variable

Automatically

They are automatically declared when first used:

```
x = 5;

y = 6;

z = x + y;
```

Variable

Using var

The var keyword was used in all JavaScript code from 1995 to 2015.

```
var x = 5;

var y = 6;

var z = x + y;
```

Variable

Using var

The var keyword was used in all JavaScript code from 1995 to 2015.

The var keyword should only be used in code written for older browsers.

```
var x = 5;

var y = 6;

var z = x + y;
```

Variable

Using let and const

The let and const keywords were added to JavaScript in 2015.

```
let x = 5;

let y = 6;

let z = x + y;


const x = 5;

const y = 6;

const z = x + y;
```

Mixed Variables

```
const price1 = 5;

const price2 = 6;

let total = price1 + price2;
```

When to Use var, let, or const?

1. Always declare variables
2. Always use const if the value should not be changed
3. Always use const if the type should not be changed (Arrays and Objects)
4. Only use let if you can't use const
5. Only use var if you MUST support old browsers.

Data Types

JavaScript has 8 Datatypes

1. **String** - is a series of characters

2. **Number** - All JavaScript numbers are stored as decimal numbers (floating point).

3. **Bigint** - All JavaScript numbers are stored in a a 64-bit floating-point format.

4. **Boolean** - can only have two values: true or false.

5. **Undefined** - variable that has not been assigned a value

6. **Null** - special value that represents an empty or unknown value

7. **Symbol** - It represents a unique identifier and can be used in various ways

8. **Object** - JavaScript objects are written with curly braces

Objects

1. Object properties are written as name:value pairs, separated by commas.
2. JavaScript objects are written with curly braces {}.

```
const person = {firstName:"John", lastName:"Doe",
age:50, eyeColor:"blue"};
```

Functions

A JavaScript function is a block of code designed to perform a particular task.

```
function myFunction(p1, p2) {

  return p1 * p2;

}
```

JavaScript Function Syntax

- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas:

  (parameter1, parameter2, ...)

- The code to be executed, by the function, is placed inside curly brackets: {}

Strings and String Methods

JavaScript strings are for storing and manipulating text.

String length
String slice()
String substring()
String substr()
String replace()
String replaceAll()
String toUpperCase()
String toLowerCase()
String concat()

String trim()
String trimStart()
String trimEnd()
String padStart()
String padEnd()
String charAt()
String charCodeAt()
String split()

JavaScript String Length

The length property returns the length of a string.

```
let text =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";

let length = text.length;
```

Extracting String Parts

- slice(start, end)
- substring(start, end)
- substr(start, length)

slice() extracts a part of a string and returns the extracted part in a new string.

```
let text = "Apple, Banana, Kiwi";

let part = text.slice(7, 13);
```

substring() is similar to slice(). The difference is that start and end values less than 0 are treated as 0 in substring().

```
let str = "Apple, Banana, Kiwi";

let part = str.substring(7, 13);
```

substr() is similar to slice(). The difference is that the second parameter specifies the length of the extracted part.

```
let str = "Apple, Banana, Kiwi";

let part = str.substr(7, 6);
```

Replacing String Content

The replace() method replaces a specified value with another value in a string:

```
let text = "Apple, Banana, Kiwi";

let newText = text.replace("Kiwi", "Mango");
```

Arrays and Array Methods

An array is a special variable, which can hold more than one value.

```
const fruits= ["Apple", "Mango",
"Papaya"];
```

An array can hold many values under a single name, and you can access the values by referring to an index number.

Creating an Array

```
const array_name = [item1, item2, ...];
```

Using the JavaScript Keyword new

It does the same with literal declaration

```
const cars = new Array("Saab", "Volvo", "BMW");
```

Array Methods

Array length
Array toString()
Array pop()
Array push()
Array shift()
Array unshift()
Array join()
Array delete()
Array concat()
Array flat()
Array splice()
Array slice()

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];

let size = fruits.length;

document.getElementById("demo").innerHTML = fruits.toString();

document.getElementById("demo").innerHTML = fruits.join(" * ");

let fruit = fruits.pop();

let length = fruits.push("Kiwi");

let fruit = fruits.shift();

fruits.unshift("Lemon");
```

```
const myGirls = ["Cecilie", "Lone"];

const myBoys = ["Emil", "Tobias", "Linus"];



const myChildren = myGirls.concat(myBoys);
```

Open Forum

PHP

PHP

Introduction

PHP is a server-side and general-purpose scripting language that is especially suited for web development.

PHP originally stood for Personal Home Page. However, now, it stands for Hypertext Preprocessor. It’s a recursive acronym because the first word itself is also an acronym.

PHP

Introduction

PHP is a server-side and general-purpose scripting language that is especially suited for web development.

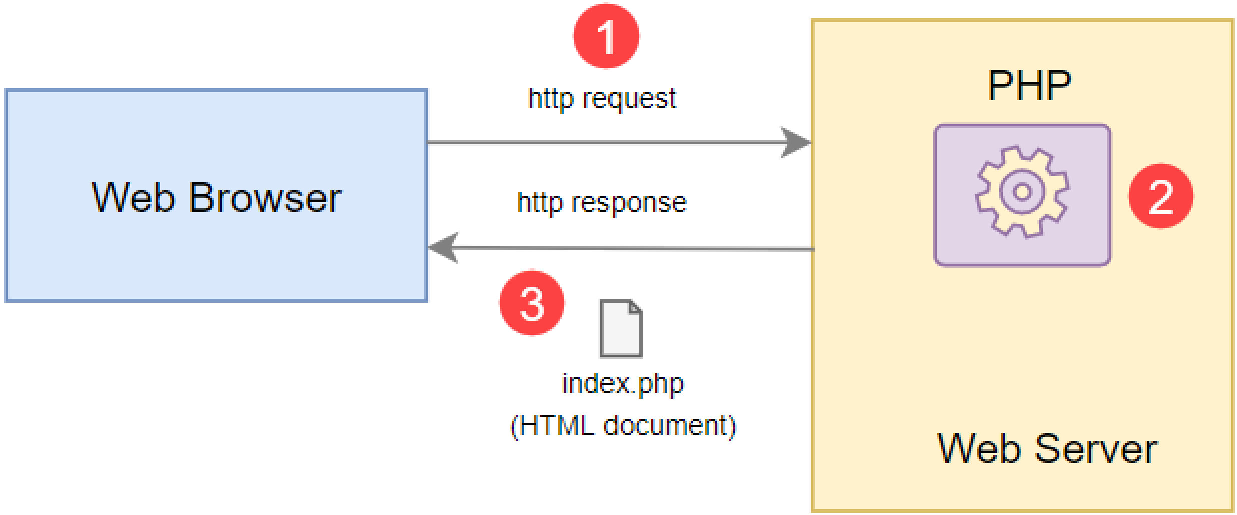
PHP originally stood for Personal Home Page. However, now, it stands for Hypertext Preprocessor. It’s a recursive acronym because the first word itself is also an acronym.

PHP Applications

PHP has two main applications:

- **Server-side scripting** – PHP is well-suited for developing dynamic websites and web applications.
- **Command-line scripting** – like Python and Perl, you can run PHP script from the command line to perform administrative tasks like sending emails and generating PDF files.

How PHP Works



XAMPP

X – [cross-platform operating systems]

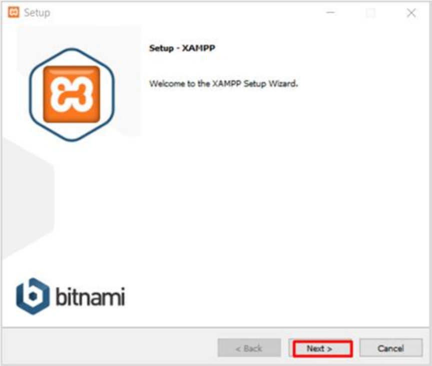
A – Apache – this is the web server software.

M – MySQL – Database.

P – PHP

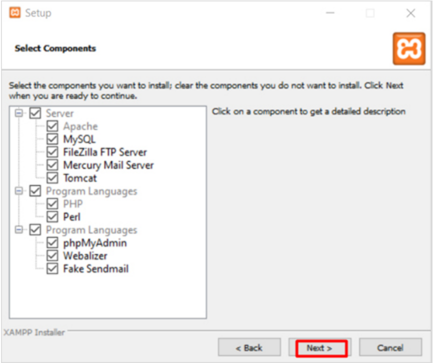
P – Perl – scripting language

Installation of PHP


Setup

Setup - XAMPP

Welcome to the XAMPP Setup Wizard.


Setup

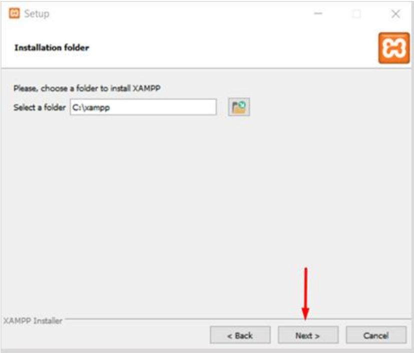
Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

Click on a component to get a detailed description

XAMPP Installer


Setup

Installation folder

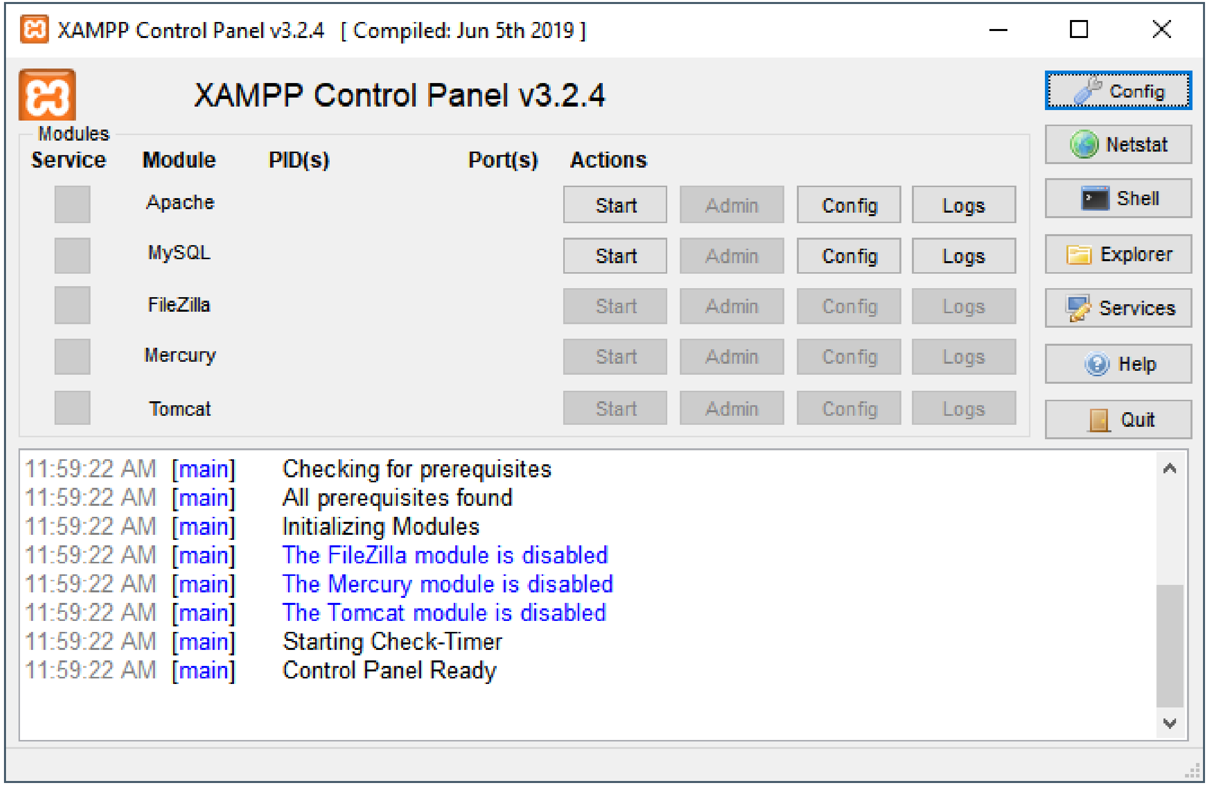
Please, choose a folder to install XAMPP

Select a folder C:\xampp

XAMPP Installer

1. Download XAMPP
2. Run and Install the XAMPP in your PC

XAMPP Control Panel

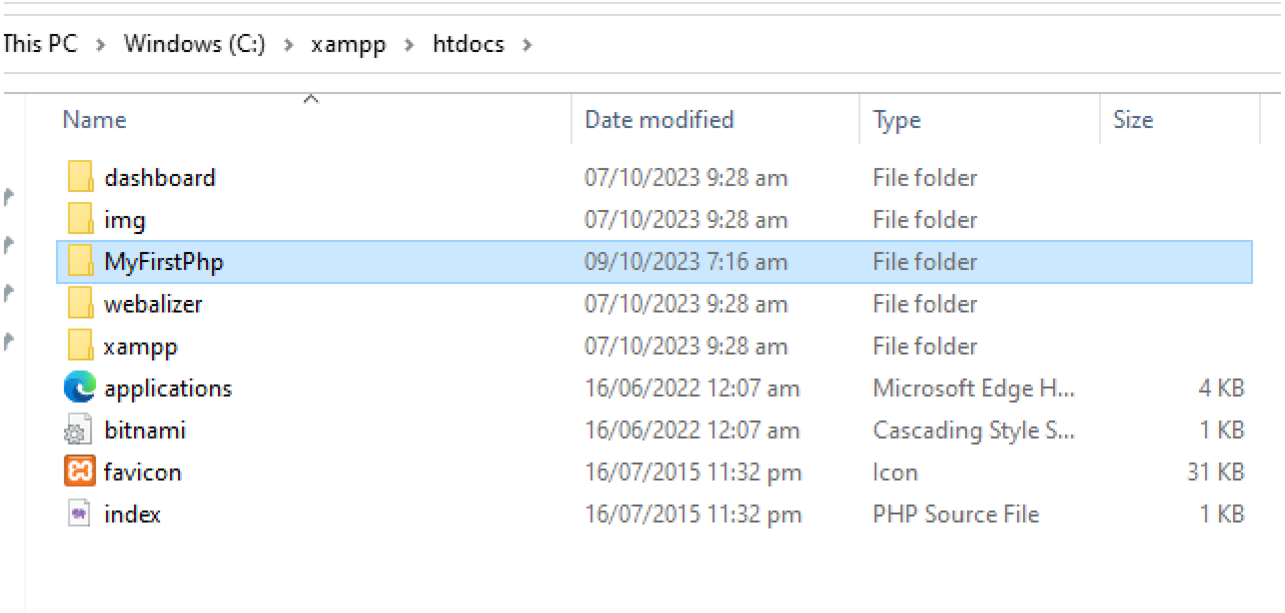


XAMPP Control Panel v3.2.4 [Compiled: Jun 5th 2019]
XAMPP Control Panel v3.2.4
Modules
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
□	Apache			Start	Admin	Config	Logs
□	MySQL			Start	Admin	Config	Logs
□	FileZilla			Start	Admin	Config	Logs
□	Mercury			Start	Admin	Config	Logs
□	Tomcat			Start	Admin	Config	Logs

```
11:59:22 AM  [main]    Checking for prerequisites
11:59:22 AM  [main]    All prerequisites found
11:59:22 AM  [main]    Initializing Modules
11:59:22 AM  [main]    The FileZilla module is disabled
11:59:22 AM  [main]    The Mercury module is disabled
11:59:22 AM  [main]    The Tomcat module is disabled
11:59:22 AM  [main]    Starting Check-Timer
11:59:22 AM  [main]    Control Panel Ready
```

Locate the htdocs folder



This PC › Windows (C:) › xampp › htdocs ›

Name	Date modified	Type	Size
dashboard	07/10/2023 9:28 am	File folder	
img	07/10/2023 9:28 am	File folder	
MyFirstPhp	09/10/2023 7:16 am	File folder	
webalizer	07/10/2023 9:28 am	File folder	
xampp	07/10/2023 9:28 am	File folder	
applications	16/06/2022 12:07 am	Microsoft Edge H...	4 KB
bitnami	16/06/2022 12:07 am	Cascading Style S...	1 KB
favicon	16/07/2015 11:32 pm	Icon	31 KB
index	16/07/2015 11:32 pm	PHP Source File	1 KB

PHP Syntax

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PHP - First PHP!</title>
</head>
<body>
        <h1><?php echo 'My First PHP!'; ?></h1>
</body>
</html>
```

The code between the opening tag <?php and closing tag ?> is PHP:

```
<?php echo 'Hello, World!'; ?>
```

PHP Variables

A variable stores a value of any type, e.g., a string, a number, an array, or an object.

A variable has a name and is associated with a value. To define a variable, you use the following syntax:

```
$variable_name = value;
```

```
<body>
         <?php
                  $title = 'PHP is
awesome!';
         ?>
         <h1>
         <?php echo $title; ?>
     </h1>
</body>
```

PHP Variables

A variable stores a value of any type, e.g., a string, a number, an array, or an object.

A variable has a name and is associated with a value. To define a variable, you use the following syntax:

```
$variable_name = value;
```

```
<body>
         <?php
                   $title = 'PHP is
awesome!';
         ?>
         <h1>
         <?php echo $title; ?>
    </h1>
</body>
```

Separation of Concern in PHP

Mixing PHP code with HTML will make the code unmaintainable, especially when the application grows. To avoid this, you can separate the code into separate files.

- **index.php** – store the logic for defining and assigning value to variables.
- **index.view.php** – store the code that displays the variables.
- Use the **require** construct to include the code from the index.view.php in the index.php file.

Sample Codes

Index.view.php

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
        <title>PHP Variables</title>
</head>
<body>
        <h1><?= $title ?></h1>
</body>
</html>
```

Index.php

```
$title = 'PHP is awesome!';

require 'index.view.php';
```

PHP Comments

Comments are important parts of the code. Comments provide useful information that will help you and other developers understand the meaning of the code more quickly later.

PHP supports two types of comments:

- **One-line comments**

  uses the // for a one-line comment:

  ```
  // single line comments

  #  single line comments
  ```

- **Multi-line comments**

  ```
  <?php

  /*

     This is an example of a multi-line
  comment,

     which can span multiple lines.

  */
  ```

Data Types

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

Data Types and Samples

PHP String

```
<?php

$x = "Hello DICT!";

echo $x;

?>
```

PHP int

```
 <?php

$x = 10.365;

var_dump($x);

?>
```

Data Types and Samples

PHP Boolean

```
<?php

$x = true;

$y = false;

echo $x;

?>
```

PHP Array

```
 <?php

$fruits =
array("Mango","Papaya","Rambutan");

var_dump($fruits);

?>
```

Object

```
 <?php
class Fruit{
  public $taste;
  public $name;
  public function __construct($taste, $name) {
           $this->taste = $taste;
           $this->name = $name;
  }
  public function message() {
           return "My favorite fruit is " .
$this->taste . " " . $this->name . "!";
  }
}

$myFavFruit = new Fruit("Sweet", "Mango");
echo $myFavFruit -> message();
echo "<br>";
$myFavFruit = new Fruit("Sour", "Tamarind");
echo $myFavFruit -> message();
?>
```

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

```
 <?php

define("GREETING", "Enjoy Coding!");

echo GREETING;

?>
```

PHP Operators

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

Arithmetic operators

Operator	Name	Example
+	Addition	$x + $y
-	Subtraction	$x - $y
*	Multiplication	$x * $y
/	Division	$x / $y
%	Modulus	$x % $y
**	Exponentiation	$x ** $y

PHP Assignment Operators

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

PHP Comparison Operators

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Increment / Decrement Operators

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

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

if-else statements and switch statement

In PHP we have the following conditional statements:

- **if statement** - executes some code if one condition is true
- **if...else statement** - executes some code if a condition is true and another code if that condition is false
- **if...elseif...else statement** - executes different codes for more than two conditions
- **switch statement** - selects one of many blocks of code to be executed

If

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

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

If/else

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

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

The PHP switch Statement

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

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

Loops

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

while

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

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

do...while Loop

loop will always execute the block of code once

```
<?php

$x = 1;

do {

                         echo "The number is: $x
<br>";

                        $x++;

} while ($x <= 5);

?>
```

for Loop

The for loop is used when you know in advance how many times the script should run.

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

foreach Loop

The foreach loop - Loops through a block of code for each element in an array.

```php
<?php

$colors = array("red", "green", "blue", "yellow");


foreach ($colors as $value) {

                                        echo "$value <br>";

}

?>
```

Functions

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

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

Create a User Defined Function

```
function functionName() {
  code to be executed;
}
```

```
 <?php

function familyName($fname, $year) {

  echo "$fname Refsnes. Born in $year
<br>";

}



familyName("Hege", "1975");

familyName("Stale", "1978");

familyName("Kai Jim", "1983");

?>
```

HTML Form

```
<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>
```

HTML Form

```
<html>
<body>

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

</body>
</html>
```