

Chapter Two

Introduction HTML



Difference Between HTML, CSS and JS



© +91-99132 99805 | www.trueinstitute.com

2024, 2027 Silver Business Plot, Near VIF Circle, Uthwara, Surat - 394105

HTML

► Hypertext Markup Language

- ◌ is the standard markup language used to create web pages.
 - HTML describes the structure of a website semantically along with cues for presentation
 - ◌ its a markup language, rather than a programming language
 - *HTML documents consist of a tree of elements and text.*
 - *Each element is denoted in the source by a start tag,*
 - ◌ *such as "<body>", and an end tag, such as "</body>".*
 - Web browsers can read HTML files and render them into visible or audible web pages.
-



Anatomy of HTML document

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is a web page.</p>
  </body>
</html>
```

- The `<!DOCTYPE html>` declaration defines this document to be HTML
- The `<html>` element is the root element of an HTML page
- The `lang` attribute defines the language of the document
- The `<meta>` element contains meta information about the document
- The `charset` attribute defines the character set used in the document
- The `<title>` element specifies a title for the document
- The `<body>` element contains the visible page content
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph



HTML Anatomy ...

- ▶ Stylesheet (CSS)

<head>

...

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

</head>

- ▶ Javascript

<head>

...

<script src="jscript.js"></script>

</head>



Tag, Attribute and Value

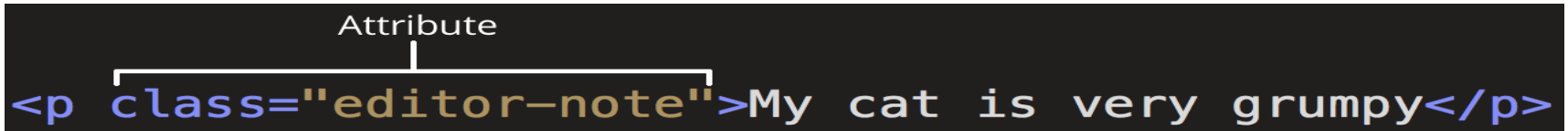
▶ Tag

▶ Void element: `<input>` `<hr>` `<br>`

▶ Non void element:

`<a>Link</a>`

▶ Attribute and Value



The diagram shows the HTML tag `<p class="editor-note">My cat is very grumpy</p>` on a dark background. A bracket above the tag is labeled "Attribute" and points to the `class="editor-note"` part. The text "My cat is very grumpy" is the content of the tag, and `</p>` is the closing tag.

- Attributes

- ⋈ Attributes are properties used to provide additional information about an element.

- ⋈ The most common attributes include *id*, *class*, *src*, *href*

- ⋈ Attributes are defined within the opening tag, after an element's name.



Tag, Attribute and Value

- ⋈ Generally attributes include a name and a value.
- ⋈ The format for these attributes consists of the attribute name followed by an equals sign and then a quoted attribute value.
- E.g.
 - ▶ `<a href="demo.html"> simple </a>`
 - ▶ `<input name="email" value="who@where.com">`
 - ▶ Attribute: name, value
 - ▶ Value: email, who@where.com



Classifying HTML elements

- **top-level elements:** html, head and body
- **head elements:** placed inside the head
 - ~ Title, style, link, meta, base, script
 - ~ Wouldn't display on the page
- **block-level elements:** flow elements that behave like paragraphs
 - ~ occupy 100% of available width
 - ~ stacked vertically with preceding and subsequent block elements
 - ~ article, h1-h6, header, footer, section, p, figure, canvas,
 - ~ pre, div, ul, ol, dl, table, form, video



Classifying HTML elements

- **inline elements**
 - ~ phrasing elements that behave like words
 - ~ flow horizontally
 - ~ usually placed inside block elements
 - ~ a (anchor), audio, br, code, img, em, nav, samp
 - ~ span, strong, sub, sup, time, var



Syntax of HTML

- All tags begin with < and end with >
- tag name is given immediately following opening <
- unrecognized tags and attributes are ignored
- attributes are given following the tag name:
 <tag attribute1="value" attribute2="value" ...>
- ' can be used instead of "
- forgetting to close a quote can result in a blank page
- It's good practice to keep HTML markup lowercase.
- values are case sensitive (used to).



Syntax of HTML

- elements without closing tags are of the form `<img />`
 - ~ ending `/` is optional for HTML5, needed for polyglot documents
- elements must be well-formed: no bad nesting
 - ~ `<p>Stuff <strong></p></strong>`
- attributes can be in any order.
- white space is allowed
 - ~ between tag name and attributes
 - ~ around the `=` sign
 - ~ within attribute value (but should be avoided)



Standard HTML Tags

- ▶ Heading

<h1>, <h2>, ... <h6>

- ▶ Table

<table>, <tr>, <th>, <td>, <tbody>, <thead>

- ▶ Paragraph

<p>

- ▶ List

**, , **

- ▶ Image

- ▶ Horizontal Line

<hr>

- ▶ Form

<form>

- ▶ Form Component

<input>, <select>, <textarea>

- ▶ Link/Anchor

<a>

- ▶ Layer Box

<div>

- ▶ Formatting

, , <sub>

- ▶ Line Break

**
**



HTML 5

- HTML5 is a cooperation between the
 - ~ World Wide Web Consortium (W3C) and
 - ~ Web Hypertext Application Technology Working Group (WHATWG).
- WHATWG was working with web forms and applications,
- W3C was working with XHTML 2.0.
- In 2006, they decided to cooperate and create a new version of HTML.



HTML 5: Ground Rules

- Some rules for HTML5 were established:
 - ⋈ New features should be based on HTML, CSS, DOM, and JavaScript
 - ⋈ Reduce the need for external plugins
 - ⋈ Better error handling
 - ⋈ More markup to replace scripting
 - ⋈ HTML5 should be device independent
 - ⋈ Dev process should be visible to the public



HTML 5: New Features

- Reduce for external plugin
- Semantic elements: header, footer, section, article, others.
- canvas for drawing
 - ~ paths of rectangles, arcs, lines, images
 - ~ mouse events
- Better support for Local offline Storage (variation of cookies)
- audio & video elements
 - ~ including drawing video on canvas
- form controls, like calendar, date, time, email, url, search, color, datetime, datetime-local, month, number, range, tel, week



New API's

- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage
- HTML Application Cache
- HTML Web Workers
- HTML SSE (Server-Sent Events)



Cont...

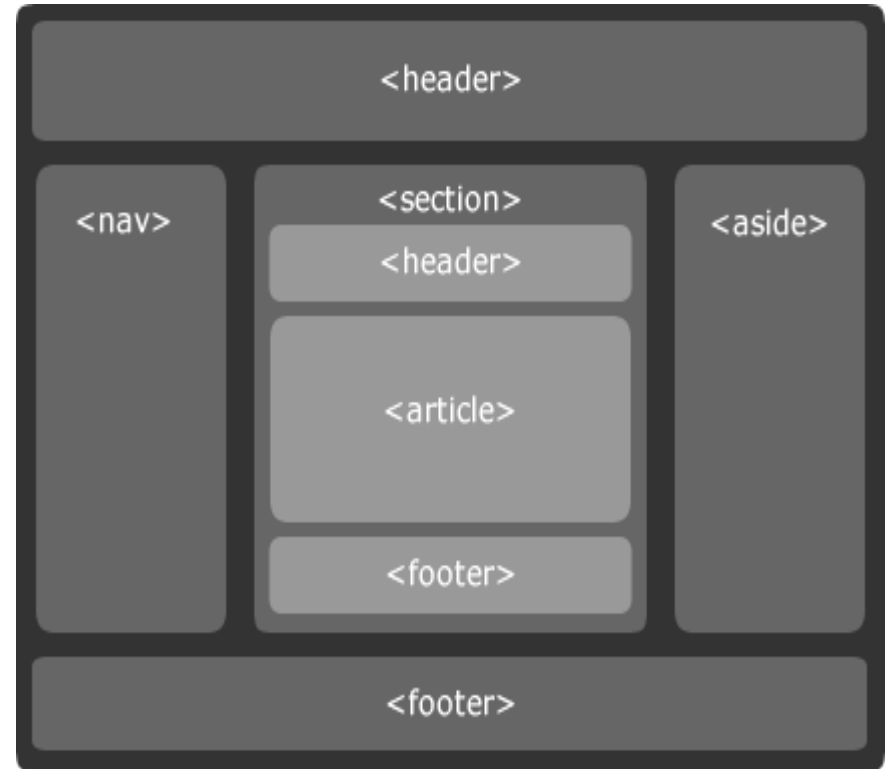
- All HTML5 elements can have the following:
 - ⌘ id – uniquely identifies an element in the page
 - ⌘ style – gives presentation style
 - ⌘ class – style class, or space-separated list of style classes
 - ⌘ title – title for the element. Can be used as tool-top display
 - ⌘ hidden – prevents element from being displayed when set to true
 - ⌘ contenteditable, draggable, dropzone, spellcheck, and more



Cont...

- HTML5 defines eight new **semantic** elements.
- All these are **block-level** elements.
- To secure correct behavior in older browsers, you can set the CSS **display** property for these HTML elements to **block**:

```
header, section, footer, aside, nav, main,  
article, figure {  
  display: block;  
}
```



HTML 5 migration

Typical HTML4

```
<div id="header">
```

```
<div id="menu">
```

```
<div id="content">
```

```
<div class="article">
```

```
<div id="footer">
```

Typical HTML5

```
<header>
```

```
<nav>
```

```
<section>
```

```
<article>
```

```
<footer>
```



Chapter 3

Introduction to CSS

CSS

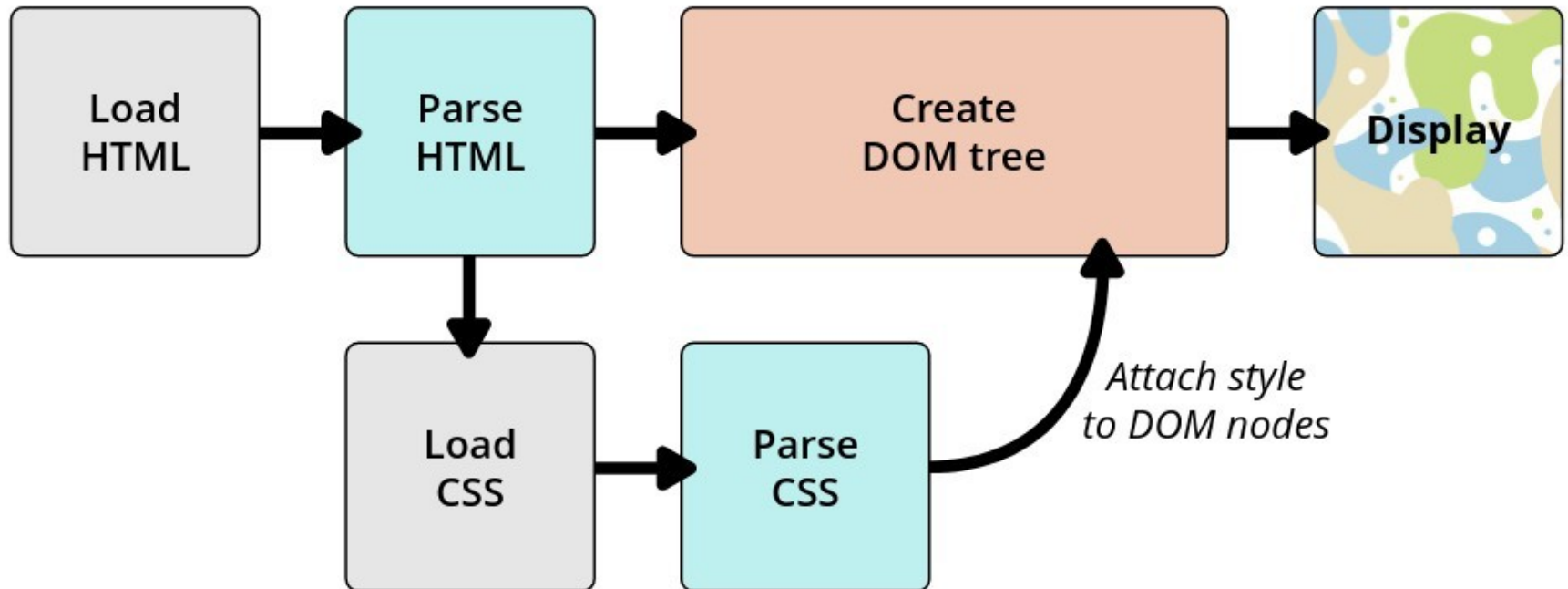


CSS

- **Cascading Style Sheets (CSS)**: is a simple mechanism for *adding style* (e.g. **fonts**, **colors**, **layouts**) to Web documents.
- Styles provide *powerful control* over the **presentation** of web pages.
- A style sheet consists of a set of **rules**.
- Each **rule** consists of one or more **selectors** and a **declaration block**.
- A **declaration block** consists of a list of **declarations** in curly braces (**{}**).
- Each **declaration** consists of a **property**, a **colon** (**:**), a **value**, then a **semi-colon** (**;**)



How CSS works



Style Sheet Syntax Explained

selector *property* *value* *rule*

```
<style type="text/css">
{
  body {
    background-color: #000000;
  }
  h1 {
    font-family: Georgia, "Times New Roman", Times, serif;
    font-size: 32px;
    color: #3099D3;
    text-align: center;
  }
</style>
```

The diagram illustrates the components of a CSS rule. Arrows point from labels to specific parts of the code: 'selector' points to 'body' and 'h1'; 'property' points to 'background-color'; 'value' points to '#000000'; and 'rule' points to the entire 'body { ... }' block. A large curly brace on the right groups the two rule blocks ('body { ... }' and 'h1 { ... }').



Creating CSS

- Create a style sheet file and apply CSS to multiple webpages, a single page, or an individual HTML element.
- Three methods:
 - External Style Sheets (preferred choice)
 - Embedded Style Sheets
 - Inline Styles (least desirable)



Example: inline

```
<!-- css.html -->

<!DOCTYPE html>
<html>
<head>
  <title>CSS Tutorial</title>

</head>
<body>

  <h3 style="color:blue"> Heading 1 </h3>
  <h3 style="color:blue"> Heading 3 </h3>
  <h3 style="color:blue"> Heading 3 </h3>

</body>
</html>
```



Example: Embedded CSS

```
<!-- css.html -->

<!DOCTYPE html>
<html>
<head>
  <title>CSS Tutorial</title>

  <style type="text/css">
    h3.h3_blue{      /*change color to blue*/
      color: blue;
    }

    h3.h3_red{       /*change color to red*/
      color:red;
    }
  </style>
</head>
<body>

  <h3 class='h3_blue'> Heading 1 </h3>
  <h3 class='h3_blue'> Heading 3 </h3>
  <h3 class='h3_blue'> Heading 3 </h3>

  <h3 class='h3_red'> Heading 1 </h3>
  <h3 class='h3_red'> Heading 3 </h3>
  <h3 class='h3_red'> Heading 3 </h3>

</body>
</html>
```



Example: External

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS Tutorial</title>
  <link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
</head>
<body>

  <h3 class='h3_blue'> Heading 1 </h3>
  <h3 class='h3_blue'> Heading 3 </h3>
  <h3 class='h3_blue'> Heading 3 </h3>

  <h3 class='h3_red'> Heading 1 </h3>
  <h3 class='h3_red'> Heading 3 </h3>
  <h3 class='h3_red'> Heading 3 </h3>

</body>
</html>
```

```
/* asset/css/my_css.css */

h3.h3_blue{
  color: blue;
}

h3.h3_red{
  color:red;
}
```



Basic CSS Selectors

- There are three types of selectors in CSS,
- **Element** : can be selected using it's name
 - ~ e.g. 'p', 'div' and 'h1' etc.
- **Class** : can be selected using '.className' operator
 - e.g. '.h3_blue'.
- **ID** : can be selected using '#idName'
 - ~ e.g. '#my_para'.



Example: Selector

```
<!-- css.html -->

<!DOCTYPE html>
<html>
<head>
  <title>CSS Selectors</title>
  <link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
</head>
<body>
  <h3>CSS Selectors</h3>

  <p class='c_head'> Paragraph with class 'c_head' </p>
  <p id='i_head'> Paragraph with id 'i_head' </p>

</body>
</html>
```

```
/* asset/css/my_css.css */

/*element selection*/
h3 {
  color: blue;
}

/*class selection*/
.c_head{
  font-family: cursive;
  color: orange;
}

/*id selection*/
#i_head{
  font-variant: small-caps;
  color: red;
}
```



More Selectors

Selectors	Description
h1, p, span etc.	element selector
.className	class selector
#idName	id selector
*	Universal selector (selects everything)
h1.className	select h1 with class 'className'
h1#idName	select h1 with id 'idName'
p span	descendant selector (select span which is inside p)
p > span	child selector ('span' which is direct descendant of 'p')
h1, h2, p	group selection (select h1, h2 and p)
span[my_id]	select 'span' with attribute 'my_id'
span[my_id=m_span]	select 'span' with attribute 'my_id=m_span'



Precedence

- Precedence
 - ⋈ **Local > Internal > External**
- Selector Priority level :
 - ⋈ **ID (highest priority)**
 - ⋈ **Class**
 - ⋈ **Element**

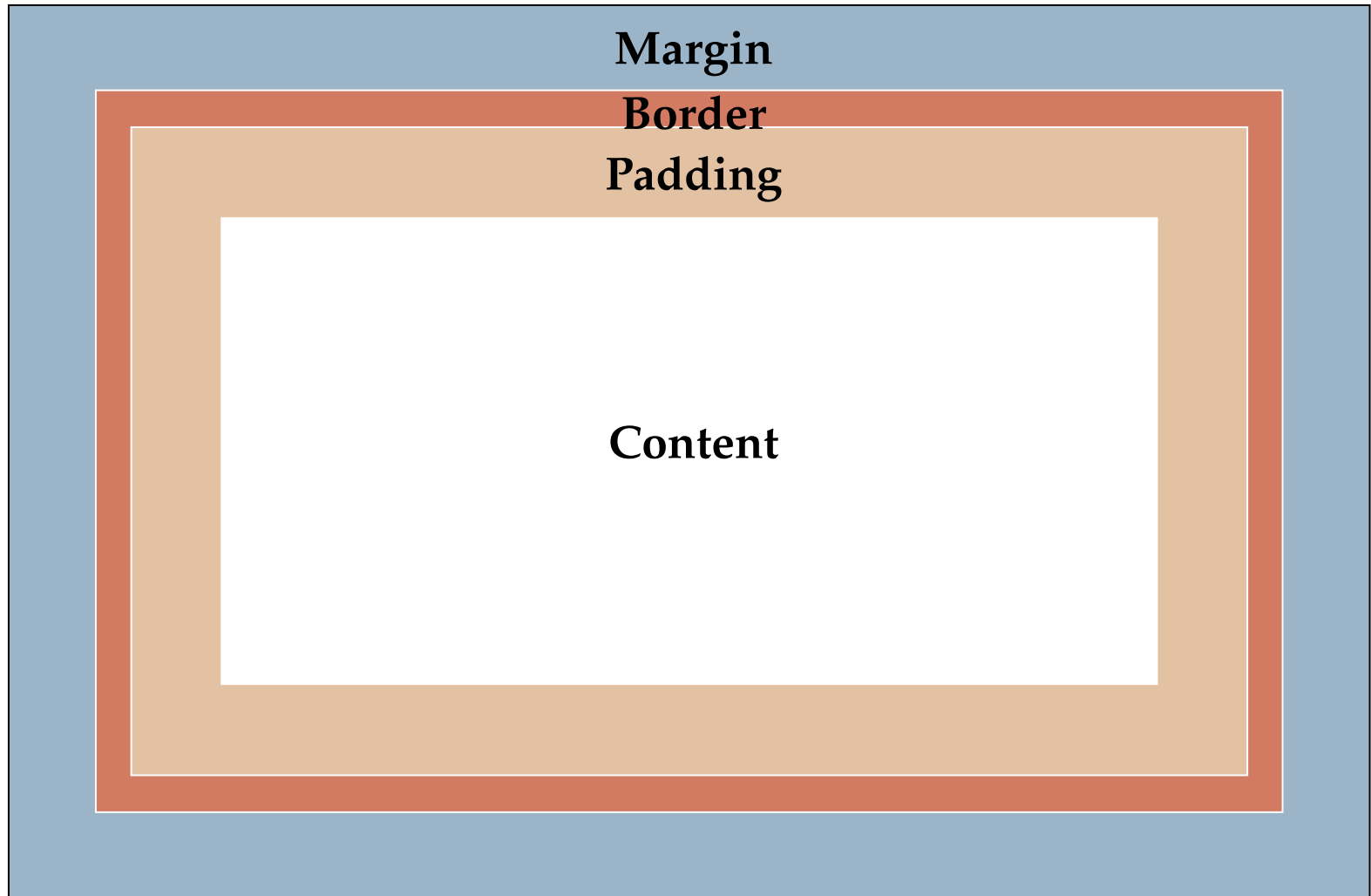


The Basis of CSS Layout

- ▶ The 3 core concepts to understand about CSS layout are:
 - ▶ The CSS box model
 - ▶ Floating
 - ▶ Positioning
- ▶ Together, these 3 concepts control the way elements are arranged and displayed on a page.



The CSS Box Model



Margins & Padding

- ▶ **Margins**
- ▶ Margins define the space around elements outside the border
- ▶ Margin properties can have negative values in order to deliberately overlap content
- ▶ Margin properties will affect the position of background elements (graphics and/or colours) in relation to the edges of the containing block element
- ▶ Margin properties can be defined independently on top, right, bottom and left, or all-at-once using CSS shorthand

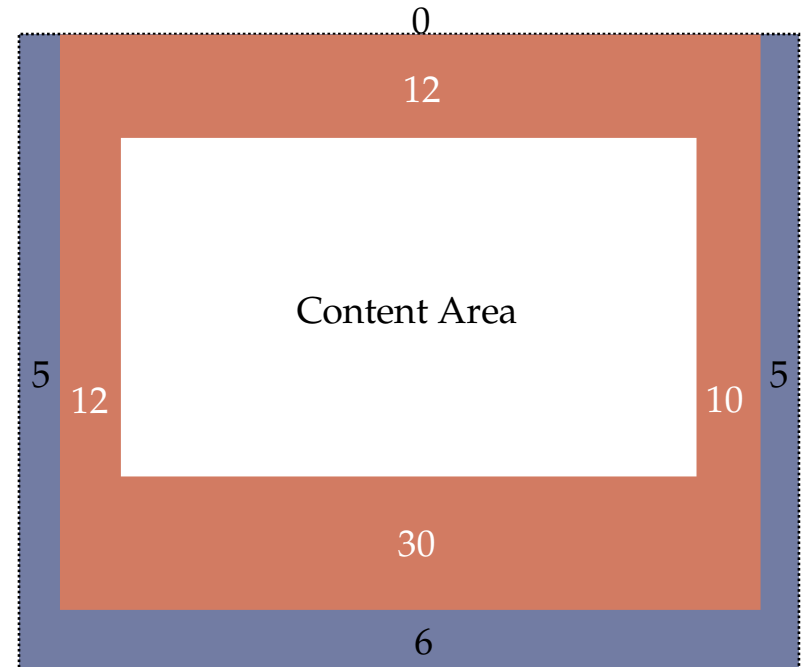
Margins & Padding

- ▶ **Padding**
- ▶ Padding defines the space around elements inside the border; i.e between the border and the content itself
- ▶ Padding properties cannot have negative values
- ▶ Padding properties do not affect the position of background elements (graphics and/or colours) in the containing block element; only the position of content.
- ▶ Padding properties can be defined independently on top, right, bottom and left, or all-at-once using CSS shorthand

CSS Shorthand: Margin & Padding

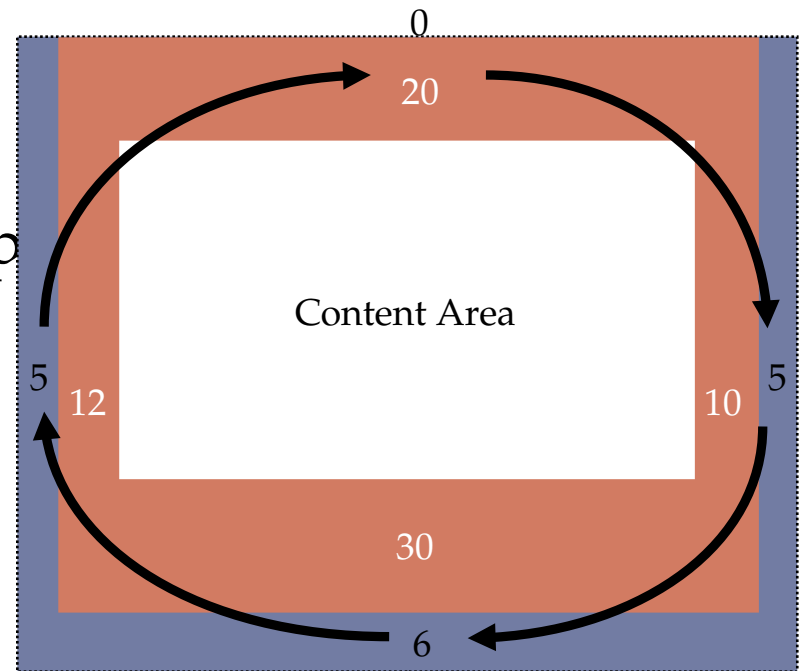
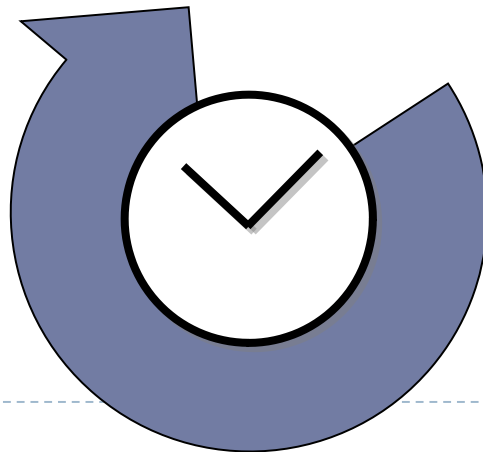
- ▶ For margin and padding (and others), CSS provides a number of shorthand properties that can save on writing lines and lines of code. Instead of writing this:

- ▶ `#container {
margin-top: 0;
margin-right: 5px;
margin-bottom: 6px;
margin-left: 5px;
padding-top: 12px;
padding-right: 10px;
padding-bottom: 30px;
padding-left: 12px;
}`



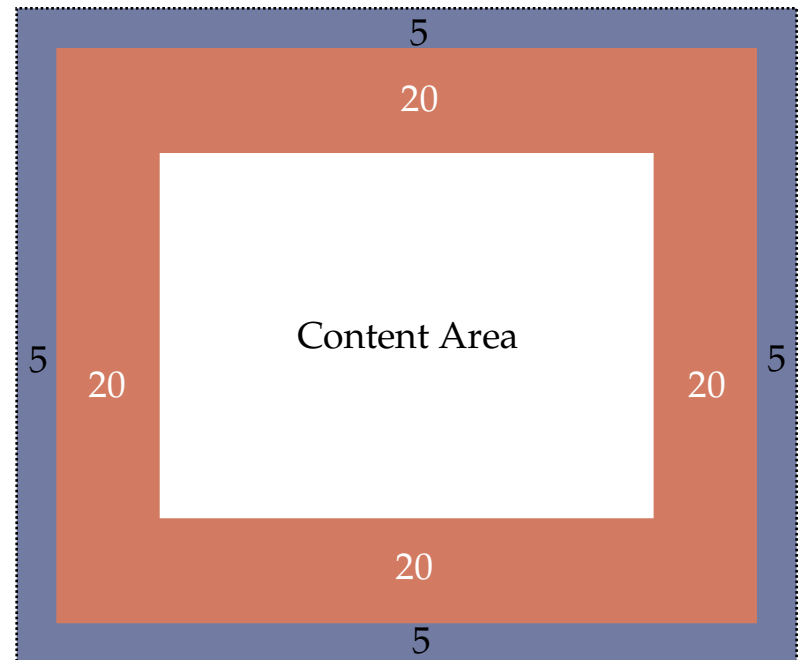
CSS Shorthand: Margin & Padding

- ▶ ...Its much easier to write this:
- ▶ `#container {
padding: 20px 10px 30px 12px;
margin: 0px 5px 6px 5px;
}`
- ▶ The sequence order is always clockwise, starting from the top



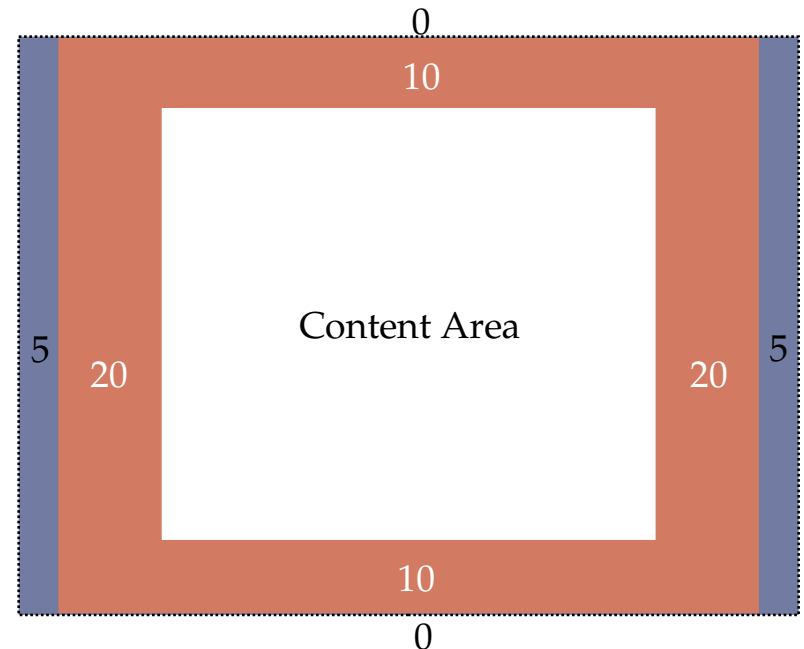
CSS Shorthand: Margin and Padding

- ▶ You can also apply just one value, example:
- ▶ `#container {
padding: 20px;
margin: 5px;
}`
- ▶ Which will apply the value specified equally on all 4 sides



CSS Shorthand: Margin and Padding

- ▶ And you can apply two values, example:
- ▶ `#container {
padding: 10px 20px;
margin: 0px 5px;
}`
- ▶ The first value is applied to the top and bottom
- ▶ The second value is applied to the left and right

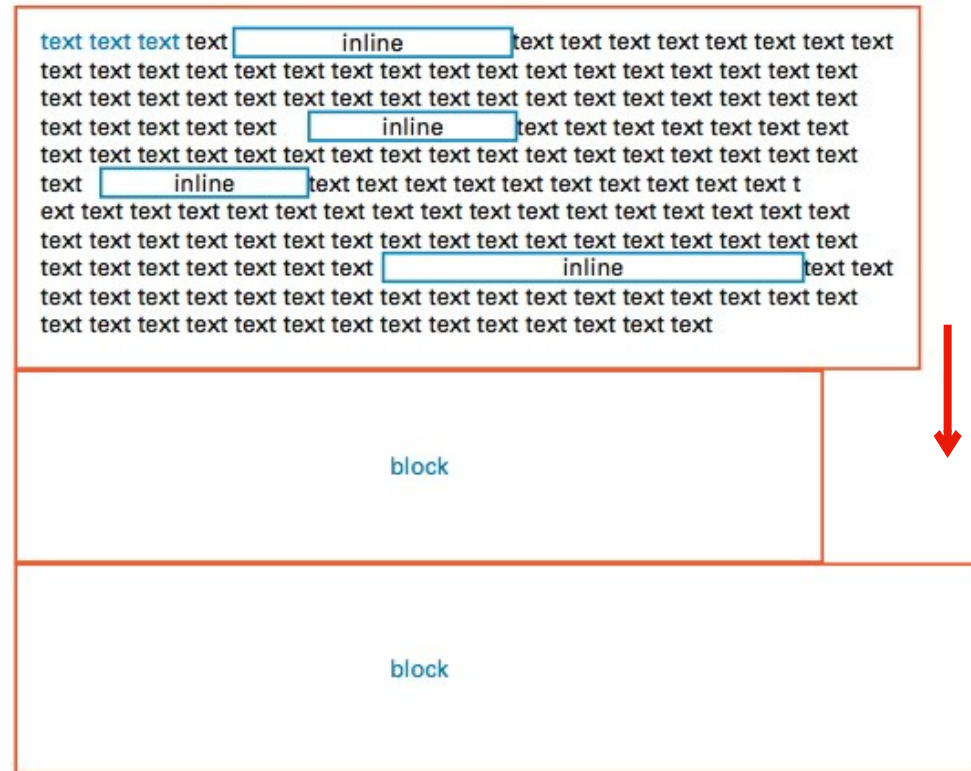


Borders

- ▶ The core border properties are:
 - ▶ **Width:** absolute (px, in, cm, or 'thin', 'medium', 'thick'), or relative (ems)
 - ▶ **Style:** dotted, dashed, solid, double, groove, ridge, inset, outset, hidden, etc
 - ▶ **Color:** 'blue', 'red', #FF9900, etc
- ▶ **Border: 5px solid blue;**

CSS Floats: “Normal Flow”

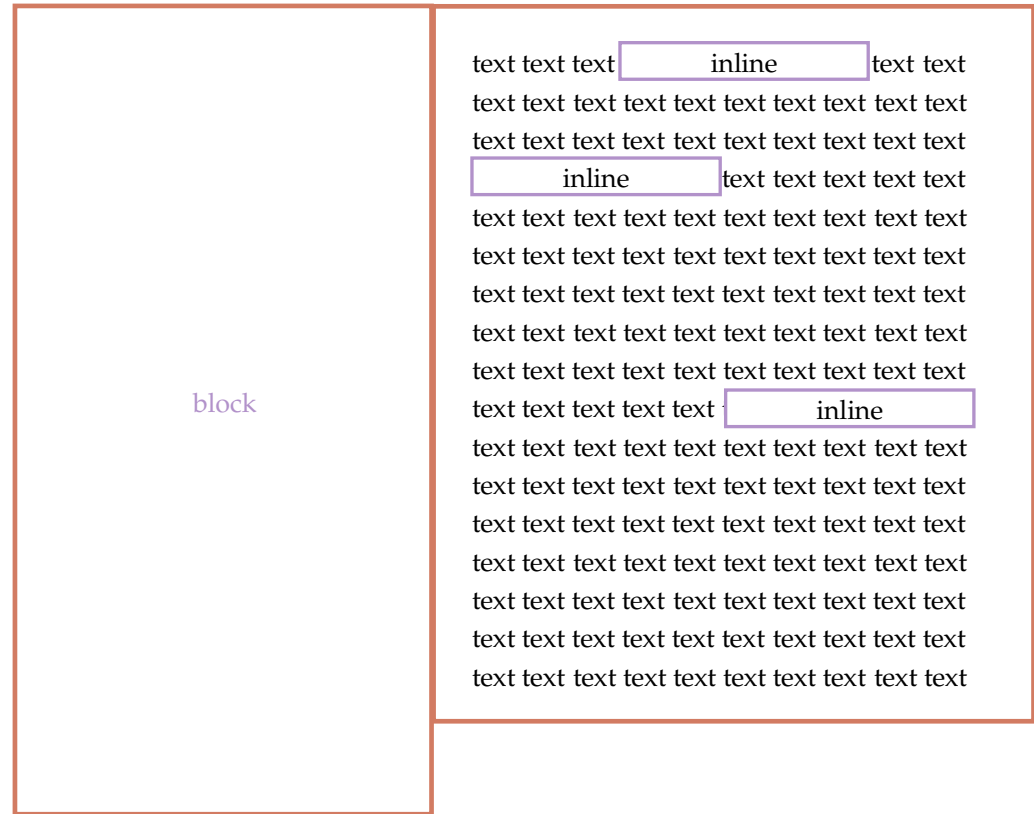
- ▶ CSS boxes for block elements are stacked, one above the other, so that they’re read from top to bottom.
- ▶ In CSS, this is said to be the “normal flow”.
- ▶ (Note that CSS boxes for inline elements are placed next to each other, within boxes for block elements, and will normally wrap according to the width of the containing block.)



But...

Floats: Positioning CSS Boxes

- ▶ ...we can position block element boxes side-by-side in CSS using floats.
- ▶ Setting the float property to left or right causes a box to be ***taken out of its position in the normal flow*** and moved as far left or right as possible.



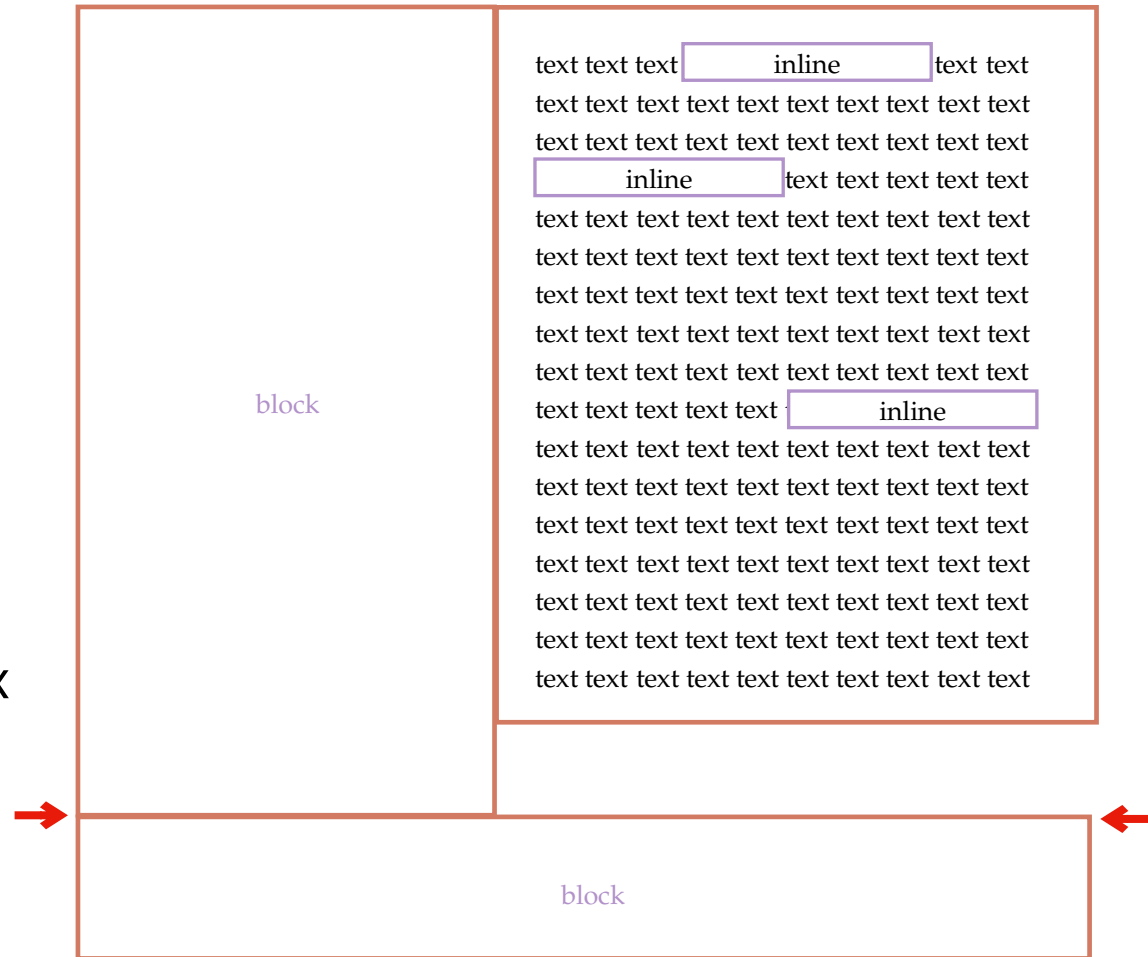
Float Values

- ▶ The Float property has three value options:
- ▶ `float: left;`
- ▶ `float: right;`
- ▶ `float: none;`



Restoring the Normal Flow: “Clear”

- ▶ To restore the “normal flow”, we can use the clear property.
- ▶ The clear property has three value options:
 - ▶ `clear: left;`
 - ▶ `clear: right;`
 - ▶ `clear: both;`
- ▶ These specify which side of the element’s box may **not** have a float next to it.



Restoring the Normal Flow: “Clear”

- ▶ Floating element taller than the container

Without Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



With Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



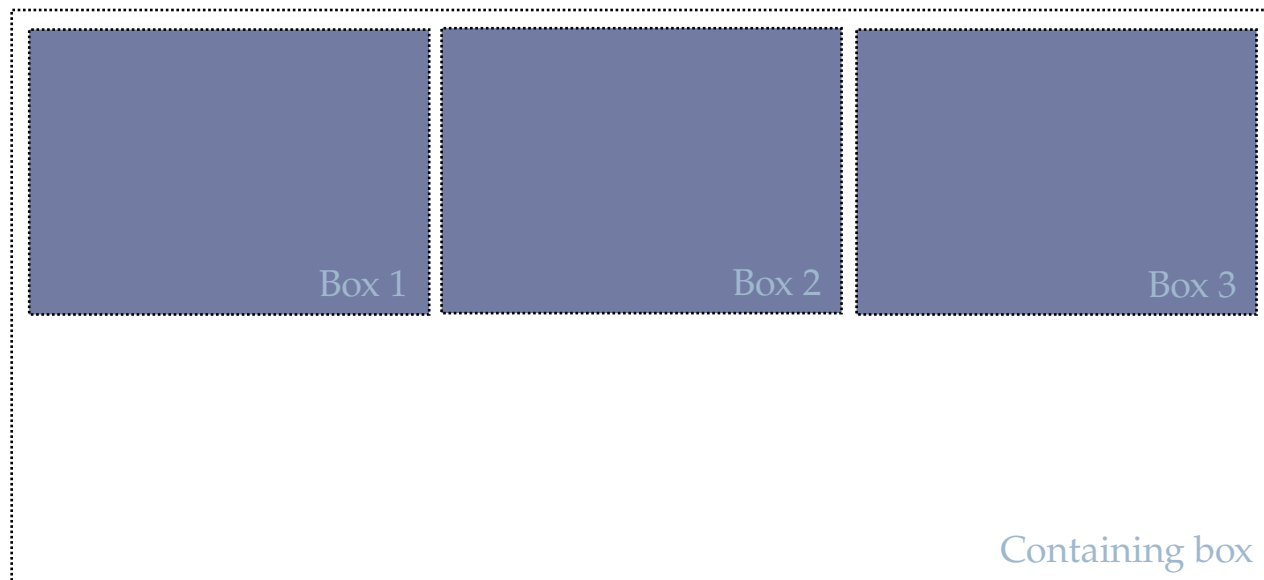
In summary, "float" is used to position elements horizontally within their container, while "clear" is used to control how elements interact with floated elements, ensuring proper layout and preventing layout issues like content overlap or misalignment.

CSS Positioning

- ▶ The third core concept to understand in CSS layout (after the 'box model' and 'floats'), is positioning.
- ▶ There are two types of positioning that can be applied to CSS boxes:
 - ▶ • **Relative Positioning**
 - ▶ • **Absolute Positioning**
- ▶ Understanding the differences between the two is difficult at first, but important!

CSS Positioning: Relative Positioning

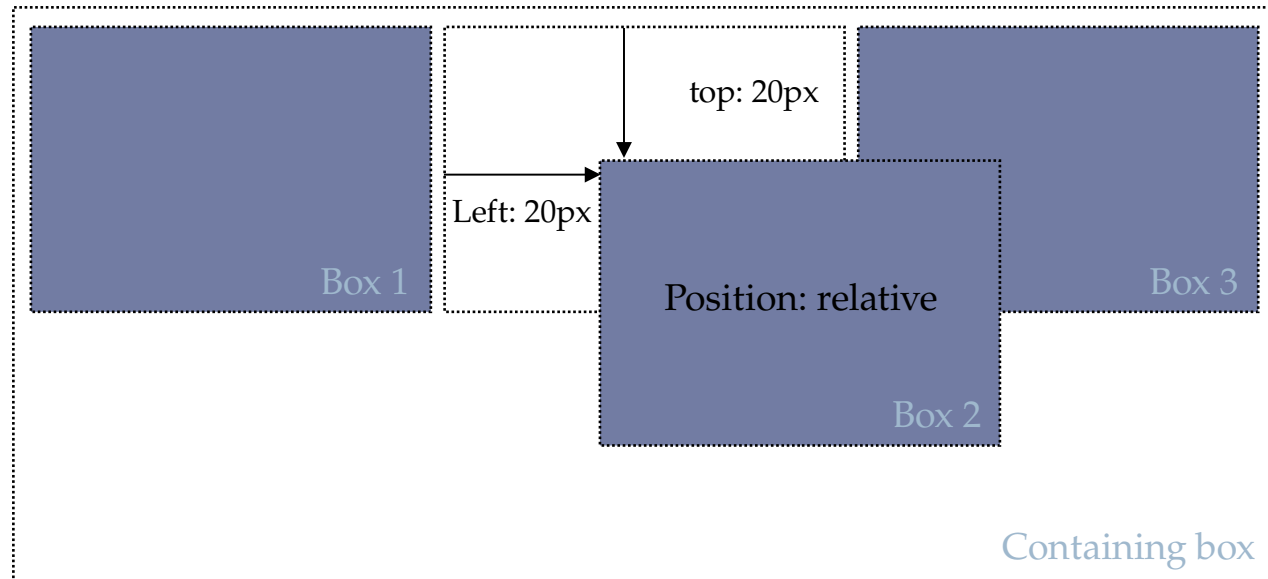
- ▶ A relatively positioned element will stay exactly where it is, in relation to the normal flow.
- ▶ You can then offset its position “relative” to its starting point in the normal flow:



CSS Positioning: Relative Positioning

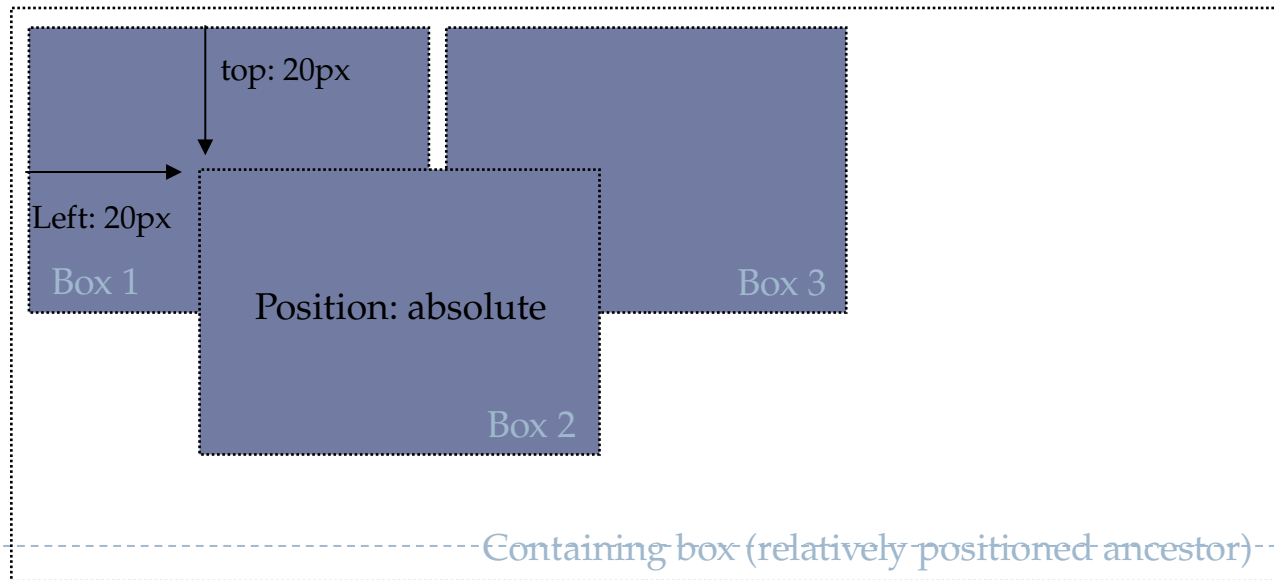
- ▶ In this example, **box 2** is offset 20px, top and left. The result is the box is offset 20px from its original position in the normal flow. Box 2 may overlap other boxes in the flow, but other boxes still recognise its original position in the flow.

```
#myBox {  
  position: relative;  
  left: 20px;  
  top: 20px;  
}
```



CSS Positioning: Absolute Positioning

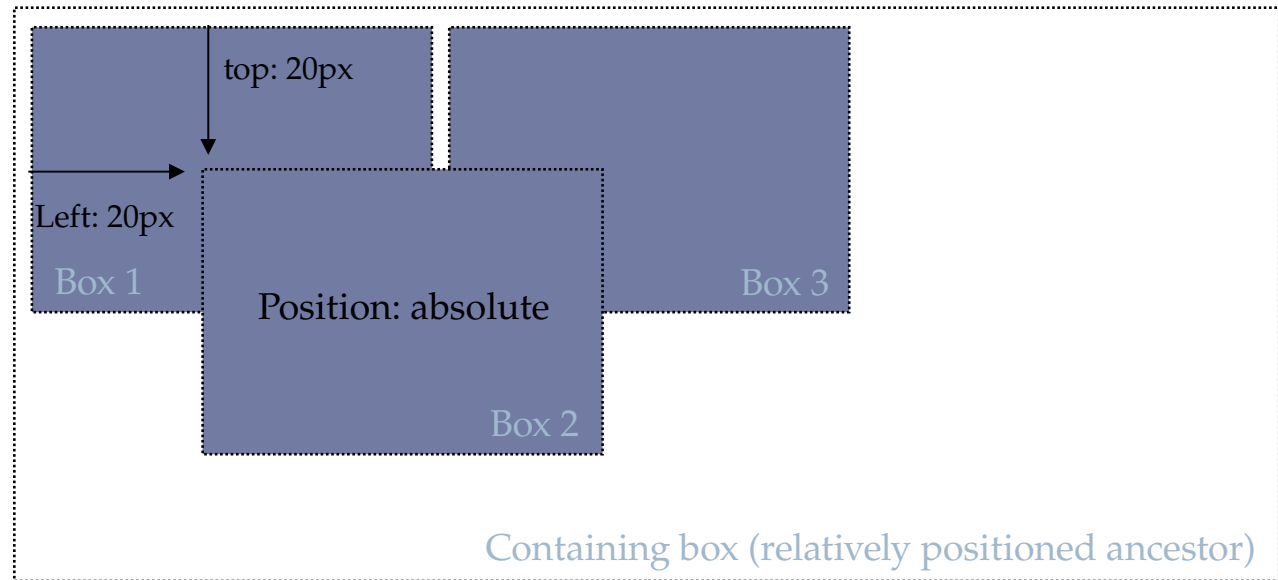
- ▶ An absolutely positioned box is taken out of the normal flow, and positioned in relation to its nearest positioned ancestor (i.e. its containing box).
- ▶ If there is no ancestor box, it will be positioned in relation to the initial containing block, usually the browser window.



CSS Positioning: Absolute Positioning

- ▶ An absolutely positioned box can be offset from its initial position inside the containing block, but other boxes within the block (and still within the normal flow) act as if the box wasn't there.

```
#myBox {  
  position: absolute;  
  left: 20px;  
  top: 20px;  
}
```



CSS Positioning: Fixed Positioning

- ▶ Fixed Positioning is a sub-category of Absolute Positioning
- ▶ Allows the creation of floating elements that are always fixed in the same position in the browser window, while the rest of the content scrolls as normal
- ▶ (rather like the effect of fixed background attachments)
- ▶ PROBLEM: fixed positioning is not supported in IE5 and IE6(!), but can be made to work with javascript for those browsers

Web-Based Information System Architecture

- ▶ **Fundamentals**
- ▶ **Categorizing Architectures**
- ▶ **Components of a Generic Web Application Architecture**
- ▶ **Layered Architectures**

Fundamentals

- ▶ A web application architecture is a structure within a website or web application to maintain the relationship between a server and user.
- ▶ It is like a skeleton of webs. When a user searches for something on the internet, this architecture responds and shows the required information from the server to the display of that device

How does this architecture work?

- ▶ Before you can grasp how a web application architecture works, you must first understand two essential components. These two components are at the heart of architecture. They are
 - ▶ Client-side: This part shows and receives the interaction of a user. It is generally written in Java, and HTML and stored in the browser. We all know it as a front-end framework.
 - ▶ Server-side: It is also known as a backend framework. Because it runs all the work in the background on HTTP. The code is generally written in PHP, Python, and Ruby.

Cont...

- ▶ Sometimes, developers use a database server to send the collected data from the browser to the server-side. Let's have a look at how this architecture works with you.
 - ▶ When you search for something, a URL request is generated.
 - ▶ The browser then searches and sends the request according to the DNS (Domain Name Server) of the URL.
 - ▶ The DNS will recognize the IP address and forward the request to the server of the URL.
 - ▶ The server then receives and sends the required information or data to the data storage. Then it locates the web page and displays that on the browser screen.

Categorizing Architectures

- ▶ Components of a Generic Web Application Architecture

Components of a Generic Web Application Architecture

- ▶ Web application architecture comes with two **types of web application architecture: Structural components, and user interface app components.**
 - **Structural Components:**

As the name suggests, these components make the structure of the application. These components include the client or web browser, database server, and web app server which are directly responsible for functions deciding the user interactions within the application. In the majority of cases, JavaScript, CSS, and HTML are used to create these components. However, it all varies with the web app developer.
 - **User Interface Components:**

The other one is the user interface components that contribute to the visual interface of the app. However, unlike structural components, that do not interact with the architecture, but are limited to displaying the web page. These components include a dashboard, widget, settings, notifications, and many other visual elements that help in making the user experience better. In other words, these components are directly responsible for the UX or the web app.

Layered Architectures

- ▶ Every **web application architecture** is built based on a layered architecture. However, it all depends on the app scale. Large applications may have four to six layers whereas small applications may have three layers. Each layer functions independently and its components are closed. Below are the **four commonest layers of web application architecture**.
 1. **Presentation Layer:**

The presentation layer aids in communication between the browser and the user interface of the application that eases the overall user interaction. Every presentation layer is created through JavaScript, HTML, CSS, and its frameworks.
 2. **Business Layer:**

The business layer helps in processing the browser requests, performs the business logic of the requests, and shares the same back to the previous layer. This layer primarily determines the business rules of the web app.
 3. **Data Access Layer:**

The data access layer is used to access data from XML, binary files, and other types of storage. In addition, it also helps in creating, reading, updating, and deleting operations.
 4. **Data Service Layer:**

The final one is the data service layer which ensures data security and stores the entire data. This layer safeguards the data by separating the app business logic from the client-side.

Reference

- ▶ <https://www.interviewbit.com/blog/web-application-architecture/>
- ▶ <https://www.thinksys.com/development/web-application-architecture-complete-guide/>
- ▶ <https://www.linkedin.com/pulse/web-application-architecture-fundamentals-benefits-best-islam/>