Applications Development and Emerging Technologies [1]

A Study Guide for Students of Sorsogon State University - Bulan Campus[2]

Jarrian Vince G. Gojar[3]

January 25, 2025

[1] A course in the Bachelor of Science in Computer Science.

[2] This book is a study guide for students of Sorsogon State University - Bulan Campus taking up the course Applications Development and Emerging Technologies.

[3] https://github.com/godkingjay

Sorsogon State University - Bulan Campus

Contents

List of Figures

List of Tables

List of Codes

Preface

–

Jarrian Vince G. Gojar
https://github.com/godkingjay

1

Introduction to Applications Development and Emerging Technologies

1.1 Introduction

1.1.1 Application

An **application** is a software program that allows users to perform specific tasks. Applications for desktop or laptop computers are sometimes called **desktop applications**, while those for mobile devices are called **mobile apps**. When you open an application, it runs inside the operating system until you close it. Most of the time, you will have more than one application open at the same time, which is known as **multitasking**.

1.1.2 Development

Development is the process of creating a software application. It includes designing the user interface, writing code, and testing the application for bugs. The goal of software development is to create a program that is easy to use and works correctly.

1.1.3 Application Development

Application development is the process of planning, designing, creating, testing, and deploying an application to perform various business operations. It can be done by massive organizations with large teams working on projects or by a single freelance developer.

1.2 Different Types of Application Development

There are several different types of application development, including:

- Web Development
- Mobile Application Development
- Desktop Application Development
- Game Development
- Cloud Development

1.2.1 Web Development

Web development is the process of creating websites and web applications. It involves designing the user interface, writing code, and testing the website for bugs. Web development can be divided into two categories: front-end development and back-end development.

1.2.1.1 Front-End Development

Front-end development is the process of creating the user interface of a website. It involves designing the layout, colors, and fonts of the website. Front-end developers use HTML, CSS, and JavaScript to create the user interface of a website.

1. **HTML (HyperText Markup Language)** is the standard markup language used to create web pages. It defines the structure of a web page using a series of elements.

2. **CSS (Cascading Style Sheets)** is a style sheet language used to define the appearance of a web page. It allows developers to control the layout, colors, and fonts of a website.

3. **JavaScript** is a programming language used to create interactive elements on a web page. It allows developers to add functionality such as animations, pop-ups, and form validation to a website.

4. **Bootstrap** is a front-end framework that allows developers to create responsive and mobile-first websites. It provides a set of pre-designed components, such as buttons, forms, and navigation bars, that can be easily customized.

5. **React** is a JavaScript library used to create user interfaces for single-page applications. It allows developers to build reusable components that update automatically when the data changes.

6. **Angular** is a front-end framework that allows developers to create dynamic web applications. It provides a set of tools and libraries for building interactive user interfaces.

7. **Vue** is a progressive JavaScript framework used to create user interfaces and single-page applications. It allows developers to build interactive web applications with ease.

The above tools and technologies are commonly used in front-end development to create responsive and interactive websites. HTML, CSS, and JavaScript are the building blocks of front-end development, while frameworks such as Bootstrap, React, Angular, and Vue provide additional features for building modern web applications.

1.2.1.2 Back-End Development

Back-end development is the process of creating the server-side logic of a website. It involves writing code that interacts with the database and processes data. Back-end developers use programming languages such as PHP, Python, and Ruby to create the server-side logic of a website.

1. **Node.js** is a JavaScript runtime environment that allows developers to run JavaScript on the server-side. It provides a set of libraries and tools for building scalable and high-performance web applications.

2. **Express** is a web application framework for Node.js. It provides a set of features for building web applications, such as routing, middleware, and templating.

3. **Django** is a high-level web framework for Python. It allows developers to build web applications quickly and efficiently. Django provides a set of tools and libraries for building secure and scalable web applications.

4. **Flask** is a lightweight web framework for Python. It allows developers to build web applications with minimal code. Flask provides a set of tools and libraries for building simple and scalable web applications.

5. **Ruby on Rails** is a web application framework for Ruby. It provides a set of tools and libraries for building web applications quickly and efficiently. Ruby on Rails follows the convention over configuration principle, which allows developers to write less code and focus on building the application.

6. **Laravel** is a web application framework for PHP. It provides a set of tools and libraries for building web applications quickly and efficiently. Laravel follows the model-view-controller (MVC) architecture, which allows developers to separate the business logic from the presentation layer.

7. **Spring** is a web application framework for Java. It provides a set of tools and libraries for building enterprise-level web applications. Spring follows the inversion of control (IoC) principle, which allows developers to write loosely coupled code and focus on building the application.

The above tools and technologies are commonly used in back-end development to create the server-side logic of a website. Back-end developers use these tools to interact with the database, process data, and handle user requests on the server-side.

1.2.2 Mobile Application Development

Mobile application development is the process of creating mobile applications for smartphones and tablets. It involves designing the user interface, writing code, and testing the mobile application for bugs. Mobile development can be divided into two categories: iOS development and Android development.

1.2.2.1 iOS Development

iOS development is the process of creating mobile applications for Apple devices, such as iPhones and iPads. It involves designing the user interface using Xcode and writing code in Swift or Objective-C. iOS developers use Xcode, Swift, and Objective-C to create mobile applications for Apple devices.

1. **Xcode** is an integrated development environment (IDE) used to create iOS applications. It provides a set of tools and libraries for building mobile applications for Apple devices.

2. **Swift** is a programming language used to create iOS applications. It provides a set of features for building mobile applications, such as type safety, optionals, and generics.

3. **Objective-C** is a programming language used to create iOS applications. It provides a set of features for building mobile applications, such as dynamic typing, message passing, and memory management.

4. **React Native** is a JavaScript framework used to create mobile applications for Android and iOS devices. It allows developers to build cross-platform mobile applications with a single codebase.

5. **Flutter** is a mobile UI framework used to create mobile applications for Android and iOS devices. It allows developers to build cross-platform mobile applications with a single codebase.

The above tools and technologies are commonly used in iOS development to create mobile applications for Apple devices. iOS developers use these tools to design the user interface and write code for mobile applications.

1.2.2.2 Android Development

Android development is the process of creating mobile applications for Android devices. It involves designing the user interface using Android Studio and writing code in Java or Kotlin. Android developers use Android Studio, Java, and Kotlin to create mobile applications for Android devices.

1. **Android Studio** is an integrated development environment (IDE) used to create Android applications. It provides a set of tools and libraries for building mobile applications for Android devices.

2. **Java** is a programming language used to create Android applications. It provides a set of features for building mobile applications, such as object-oriented programming, inheritance, and polymorphism.

3. **Kotlin** is a programming language used to create Android applications. It provides a set of features for building mobile applications, such as null safety, extension functions, and coroutines.

4. **React Native** is a JavaScript framework used to create mobile applications for Android and iOS devices. It allows developers to build cross-platform mobile applications with a single codebase.

5. **Flutter** is a mobile UI framework used to create mobile applications for Android and iOS devices. It allows developers to build cross-platform mobile applications with a single codebase.

The above tools and technologies are commonly used in Android development to create mobile applications for Android devices. Some of the tools here are also used in iOS development to create mobile applications for Apple devices. React Native and Flutter in particular are used to build cross-platform mobile applications for both Android and iOS devices.

1.2.3 Desktop Application Development

Desktop application development is the process of creating desktop applications for Windows, macOS, and Linux. It involves designing the user interface, writing code, and testing the desktop application for bugs.

1. **Electron** is a framework used to create desktop applications with web technologies. It allows developers to build cross-platform desktop applications with HTML, CSS, and JavaScript.

2. **JavaFX** is a framework used to create desktop applications with Java. It provides a set of tools and libraries for building cross-platform desktop applications with Java.

3. **Qt** is a framework used to create desktop applications with C++. It provides a set of tools and libraries for building cross-platform desktop applications with C++.

4. **WinForms** is a framework used to create desktop applications with C#. It provides a set of tools and libraries for building desktop applications for Windows.

5. **WPF** is a framework used to create desktop applications with C#. It provides a set of tools and libraries for building desktop applications for Windows.

The above tools and technologies are commonly used in desktop development to create desktop applications for Windows, macOS, and Linux. For Windows, developers use WinForms and WPF to create desktop applications with C#. For cross-platform desktop applications, developers use Electron, JavaFX, and Qt to build desktop applications with web technologies, Java, and C++.

1.2.4 Game Development

Game development is the process of creating video games for consoles, computers, and mobile devices. It involves designing the gameplay, writing code, and testing the game for bugs.

1. **Unity** is a game engine used to create 2D and 3D games for consoles, computers, and mobile devices. It provides a set of tools and libraries for building cross-platform games with C#.

2. **Unreal Engine** is a game engine used to create 2D and 3D games for consoles, computers, and mobile devices. It provides a set of tools and libraries for building cross-platform games with C++.

3. **Godot** is a game engine used to create 2D and 3D games for consoles, computers, and mobile devices. It provides a set of tools and libraries for building cross-platform games with GDScript.

4. **GameMaker Studio** is a game engine used to create 2D games for consoles, computers, and mobile devices. It provides a set of tools and libraries for building cross-platform games with GML.

5. **Construct** is a game engine used to create 2D games for consoles, computers, and mobile devices. It provides a set of tools and libraries for building cross-platform games with events.

The above tools and technologies are commonly used in game development to create video games for consoles, computers, and mobile devices. Unity, Unreal Engine, Godot, GameMaker Studio, and Construct are popular game engines used by game developers to create 2D and 3D games. These game engines provide a set of tools and libraries for building cross-platform games with C#, C++, GDScript, and GML.

1.2.5 Cloud Development

Cloud development is the process of creating cloud-based applications that run on remote servers. It involves designing the user interface, writing code, and testing the cloud application for bugs.

1. **Amazon Web Services (AWS)** is a cloud platform used to create cloud-based applications. It was one of the first cloud platforms and is widely used by developers to build scalable and secure cloud applications. It is a subsidiary of Amazon providing on-demand cloud computing platforms and APIs to individuals,

2. **Microsoft Azure**, similarly to AWS, is a cloud platform used to create cloud-based applications. Microsoft Azure is a cloud computing service created by Microsoft for

building, testing, deploying, and managing applications and services through Microsoft-managed data centers.

3. **Google Cloud Platform (GCP)**, similarly to AWS and Microsoft Azure, is a cloud platform used to create cloud-based applications. Google Cloud Platform is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube.

4. **Heroku** is a cloud platform used to create cloud-based applications. It provides a set of tools and services for building scalable and secure cloud applications. Heroku is a cloud platform as a service supporting several programming languages.

5. **Firebase**, also developed by Google, is a cloud platform used to create cloud-based applications. Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development.

The above tools and technologies are commonly used in cloud development to create cloud-based applications that run on remote servers. AWS, Microsoft Azure, GCP, Heroku, and Firebase are popular cloud platforms used by developers to build scalable and secure cloud applications. These cloud platforms provide a set of tools and services for building cloud-based applications with ease.

2

Web Development

2.1 Introduction

There are around 3.58 billion internet users on the planet. This implies that over half of the world's 7.6 billion people have access to the internet, which they use for everything from entertainment to education, communication to commerce, keeping up with current events, and keeping up with business experts. Indeed, for most people, the internet is the first (and often only) channel through which we communicate with the world in all of its complexities.

There are three interactive elements on the internet:

1. **Websites** - A collection of web pages that are linked together and share a common domain name.
2. **Servers** - A computer or computer program that manages access to a centralized resource or service in a network.
3. **Browsers** - A software application used to access and view websites on the internet.

The frontend (client side) and the backend (server side) are two parts of any website. The frontend comprises everything the user sees and experiences instantly while visiting a website. The backend is behind the scenes that store, send and receive information.

HTML, CSS, and Javascript files make up everything a user sees on a website. As a web developer, these are the most basic tools needed. They are the languages that required to build websites.

2.2 HTML

HTML (HyperText Markup Language) is the standard markup language used to create web pages. It defines the structure of a web page using a series of elements. It contains the essential elements of a website, such as words, titles, and paragraphs, as well as links, images, and other media. HTML elements are represented by tags, which are enclosed in angle brackets. HTML forms the backbone of any webpage, dictating its structure and content.

```
1 <!DOCTYPE html>
2 <html lang="en">
3
```

```
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>First Web Page</title>
</head>

<body>
    <h1>Hello, World!</h1>
    <p>Welcome to my website.</p>
</body>

</html>
```

Code 2.1: HTML Example

Code 2.1 shows an example of an HTML document. An HTML boilerplate usually looks like this:

- **<!DOCTYPE html>** - Defines the document type and version of HTML. In this case, it is HTML5.
- **<html>** - Defines the root element of an HTML page.
- **<head>** - Contains meta-information about the document, such as the title, character set, and viewport.
- **<body>** - Contains the content of the document, such as headings, paragraphs, and images.

2.2.1 HTML Tags

HTML tags are used to define the structure and content of a web page. They are enclosed in angle brackets and come in pairs: an opening tag and a closing tag. The opening tag is used to define the beginning of an element, while the closing tag is used to define the end of an element.

When an HTML tag is opened, it must be closed to avoid errors. Some tags are self-closing, meaning they do not require a closing tag. HTML tags can also have attributes, which provide additional information about the element.

```
<p>Welcome to my website.</p>
```

Code 2.2: HTML Open and Close Tag

Code 2.2 shows an example of an HTML tag with an opening tag (** **) and a closing tag (** **). The content of the paragraph is "Welcome to my website.".

```
<img src="https://avatar.iran.liara.run/public" alt="Image" />
```

Code 2.3: HTML Self-Closing Tag

Code 2.3 shows an example of an HTML tag that is self-closing (**<img />**). This tag is used to insert an image into the document. The **src** attribute specifies the URL of the image, while the **alt** attribute provides a text description of the image.

2.2.1.1 <html>...</html>

This tag specifies that the webpage is written in HTML. It appears at the very first and last line of the webpage. It is mainly used to show that the page uses HTML5 – the latest version of the language. Also known as the root element, this tag can be thought of as a parent tag for every other tag used in the page.

```
<!DOCTYPE html>
<html lang="en">
    <!-- Content goes here -->
</html>
```

Code 2.4: HTML <html> Tag

Code 2.4 shows an example of the <html> tag. Here, the **lang** attribute specifies the language of the document, which is English in this case.

2.2.1.2 <head>...</head>

This tag is used to define the head section of the webpage. The head section contains meta-information about the document, such as the title, character set, and viewport. It is not displayed on the webpage but is used to provide information about the document to the browser and search engines.

```
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>First Web Page</title>
</head>
```

Code 2.5: HTML <head> Tag

Code 2.5 shows an example of the <head> tag. Here, the **<meta>** tag is used to define the character set and viewport of the document, while the **<title>** tag is used to define the title of the document. This title appears in the browser tab.

2.2.1.3 <title>...</title>

This tag is used to define the title of the document. It appears in the browser tab and is used to identify the webpage.

```
<title>First Web Page</title>
```

Code 2.6: HTML <title> Tag

Code 2.6 shows an example of the <title> tag. Here, the title of the document is "First Web Page". This title appears in the browser tab when the document is opened.

2.2.1.4 <body>...</body>

This tag is used to define the body section of the webpage. The body section contains the content of the document, such as headings, paragraphs, and images. It is displayed on the webpage and is visible to the user.

```
<body>
    <h1>Hello, World!</h1>
    <p>Welcome to my website.</p>
</body>
```

Code 2.7: HTML <body> Tag

Code 2.7 shows an example of the <body> tag. Here, the **<h1>** tag is used to define a heading, while the ** ** tag is used to define a paragraph. This content is displayed on the webpage and is visible to the user.

2.2.1.5 <h1>...</h1> to <h6>...</h6>

These tags are used to define headings of different sizes. The **<h1>** tag defines the largest heading, while the **<h6>** tag defines the smallest heading. Headings are used to define the structure of the document and provide a hierarchy of information.

```
<h1>This is a Heading 1</h1>
<h2>This is a Heading 2</h2>
<h3>This is a Heading 3</h3>
<h4>This is a Heading 4</h4>
<h5>This is a Heading 5</h5>
<h6>This is a Heading 6</h6>
```

Code 2.8: HTML <h1> to <h6> Tags

Code 2.8 shows an example of the **<h1>** to **<h6>** tags. These tags are used to define headings of different sizes, with **<h1>** being the largest and **<h6>** being the smallest.

2.2.1.6 ...

This tag is used to define a paragraph of text. It is used to group text content together and provide structure to the document.

```
<p>
  Excepteur officia tempor do laborum commodo cupidatat ea Lorem qui irure
      enim velit. Adipisicing dolor minim Lorem nulla dolor quis et aliqua.
      Officia anim adipisicing excepteur sint elit qui laboris reprehenderit
      non elit. Voluptate voluptate duis aliqua proident elit exercitation
      cillum anim reprehenderit nostrud minim culpa veniam.
</p>
```

Code 2.9: HTML Tag

Code 2.9 shows an example of the **<p>** tag. This tag is used to define a paragraph of text, which is displayed on the webpage.

2.2.1.7 ...

This tag is used to define a division or section of the document. It is a block-level element that can contain other block-level or inline elements.

```
<div>
    <h1>Hello, World!</h1>
    <p>Welcome to my website.</p>
</div>

<div>
    <img src="https://avatar.iran.liara.run/public/boy" alt="Image" />
    <img src="https://avatar.iran.liara.run/public/girl" alt="Image" />
</div>
```

Code 2.10: HTML Tag

Code 2.10 shows an example of the ** ** tag. This tag is used to define a division or section of the document, which can contain other elements such as headings, paragraphs, and images.

2.2.1.8 <section>...</section>

This tag is used to define a section of the document. It is usually used to group related content together such as articles, blog posts, or product listings.

```
<section>
    <h2>Section 1</h2>
    <p>Content for section 1 goes here.</p>
</section>

<section>
    <h2>Section 2</h2>
    <p>Content for section 2 goes here.</p>
</section>

<section>
    <h2>Section 3</h2>
    <p>Content for section 3 goes here.</p>
</section>
```

Code 2.11: HTML <section> Tag

Code 2.11 shows an example of the **<section>** tag. This tag is used to define a section of the document, which can contain related content such as headings and paragraphs.

2.2.1.9 ...

This tag is used to define a span of text. It is an inline element that can contain other inline elements. It is usually used to apply styles to a specific section of text.

```
<p>Welcome to my <span style="color: blue;">website</span>.</p>
```

Code 2.12: HTML Tag

Code 2.12 shows an example of the **** tag. This tag is used to define a span of text, which can be styled separately from the rest of the paragraph. In this example, the text "website" is styled with a blue color.

2.2.1.10

This tag is used to insert a line break in the document. It is a self-closing tag that does not require a closing tag. When used, it moves the content to the next line. In texts, it is used to separate paragraphs or lines.

```
<p>Welcome to my <br /> website.</p>
```

Code 2.13: HTML
 Tag

Code 2.13 shows an example of the **
** tag. This tag is used to insert a line break in the document, which moves the content to the next line.

2.2.1.11 <hr />

This tag is used to insert a horizontal rule in the document. It is a self-closing tag that creates a horizontal line across the page. It can be used to separate sections of content or to create a visual break in the document.

```
<p>Welcome to my website.</p>
<hr />
<p>Thank you for visiting.</p>
```

Code 2.14: HTML <hr /> Tag

Code 2.14 shows an example of the **<hr />** tag. This tag is used to insert a horizontal rule in the document, which creates a horizontal line across the page.

2.2.1.12 <img />

This tag is used to insert an image in the document. It is a self-closing tag that requires the **src** attribute to specify the image file.

```
<img src="https://avatar.iran.liara.run/public" alt="Image" />
```

Code 2.15: HTML <img /> Tag

Code 2.15 shows an example of the **<img />** tag. This tag is used to insert an image in the document, which is displayed on the webpage. The **src** attribute specifies the image file, while the **alt** attribute provides alternative text for the image.

2.2.1.13 <a>...

This tag is used to create a hyperlink in the document. It requires the **href** attribute to specify the URL of the link.

```
<a href="https://www.github.com/godkingjay" target="_blank>Visit GitHub</a>
```

Code 2.16: HTML <a> Tag

Code 2.16 shows an example of the **<a>** tag. This tag is used to create a hyperlink in the document, which links to the specified URL. The **href** attribute specifies the URL of the link. The **target** attribute specifies where to open the link. Its value can be **_blank** to open the link in a new tab or **_self** to open the link in the same tab. By default, the link opens in the same tab.

2.2.1.14 <ul>...

This tag is used to create an unordered list in the document. It contains a list of items that are displayed with bullet points. In the list, each item is defined using the **<li>** tag.

```
<ul style="list-style-type: square;">
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

Code 2.17: HTML <ul> Tag

Code 2.17 shows an example of the **<ul>** tag. This tag is used to create an unordered list in the document, which contains a list of items displayed with bullet points. The **<li>** tag is used to define each item in the list. The **style** attribute is used to specify the style of the list, such as the type of bullet point. The **list-style-type** property specifies the type of bullet point to use, such as *square*, *circle*, or *disc*.

2.2.1.15 <ol>...

This tag is used to create an ordered list in the document. It contains a list of items that are displayed with numbers or letters.

```
<ol type="A" start="3">
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ol>
```

Code 2.18: HTML <ol> Tag

Code 2.18 shows an example of the **<ol>** tag. This tag is used to create an ordered list in the document, which contains a list of items displayed with numbers or letters. The **<li>** tag is used to define each item in the list. The **type** attribute is used to specify the type of numbering to use, such as *1*, *A*, *a*, *I*, or *i*. The default type is *1*. The **start** attribute is used to specify the starting number of the list. The default start number is *1*.

2.2.1.16 <li>...

This tag is used to define an item in a list. It is used inside the **<ul>** or **<ol>** tag to define each item in the list.

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>

<ol>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ol>
```

Code 2.19: HTML <li> Tag

Code 2.19 shows an example of the **<li>** tag. This tag is used to define an item in a list, which is displayed as part of an unordered or ordered list.

2.2.1.17 ...

This tag is used to define text that should be displayed in a strong or bold font. It is used to emphasize important text content.

```
<p>Welcome to my <strong>website</strong>.</p>
```

Code 2.20: HTML Tag

Code 2.20 shows an example of the **** tag. This tag is used to define text that should be displayed in a strong or bold font, which emphasizes the importance of the text content.

2.2.1.18 ...

Similar to the **** tag, this tag is used to define text that should be displayed in a bold font. It is used to emphasize important text content.

```
<p>Welcome to my <b>website</b>.</p>
```

Code 2.21: HTML Tag

Code 2.21 shows an example of the **<b>** tag. This tag is used to define text that should be displayed in a bold font, which emphasizes the importance of the text content.

2.2.1.19 ...

This is another inline element that is used to define text that should be displayed in an emphasized or italic font. It is used to provide emphasis to text content.

```
<p>Welcome to my <em>website</em>.</p>
```

Code 2.22: HTML Tag

Code 2.22 shows an example of the **** tag. This tag is used to define text that should be displayed in an emphasized or italic font, which provides emphasis to the text content.

2.2.1.20 ...

Similar to the **** tag, this tag is used to define text that should be displayed in an italic font. It is used to provide emphasis to text content.

```
<p>Welcome to my <i>website</i>.</p>
```

Code 2.23: HTML Tag

Code 2.23 shows an example of the **** tag. This tag is used to define text that should be displayed in an italic font, which provides emphasis to the text content.

2.2.1.21 <table>...</table>

This tag is used to create a table in the document. It contains a set of rows and columns that display data in a structured format.

```
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>John</td>
        <td>25</td>
    </tr>
    <tr>
        <td>Jane</td>
        <td>30</td>
    </tr>
</table>
```

Code 2.24: HTML <table> Tag

Code 2.24 shows an example of the **<table>** tag. This tag is used to create a table in the document, which contains a set of rows and columns that display data in a structured format. The **<tr>** tag is used to define a row in the table, while the **<th>** tag is used to define a header cell and the **<td>** tag is used to define a data cell. The **border** attribute is used to specify the border width of the table.

2.2.1.22 <tr>...</tr>

This tag is used to define a row in a table. It is used inside the **<table>** tag to define each row in the table.

```
<table border="1>
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>John</td>
        <td>25</td>
    </tr>
    <tr>
        <td>Jane</td>
        <td>30</td>
    </tr>
</table>
```

Code 2.25: HTML <tr> Tag

Code 2.25 shows an example of the **<tr>** tag. This tag is used to define a row in a table, which contains a set of cells that display data in a structured format.

2.2.1.23 <th>...</th> and <td>...</td>

These tags are used to define header cells and data cells in a table, respectively. The **<th>** tag is used to define a header cell, while the **<td>** tag is used to define a data cell.

```
<table border="1>
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>John</td>
        <td>25</td>
    </tr>
    <tr>
        <td>Jane</td>
        <td>30</td>
    </tr>
</table>
```

Code 2.26: HTML <th> and <td> Tags

Code 2.26 shows an example of the **<th>** and **<td>** tags. The **<th>** tag is used to define a header cell in a table, while the **<td>** tag is used to define a data cell in a table.

2.2.1.24 <form>...</form>

This tag is used to create a form in the document. It contains a set of form elements, such as input fields, buttons, and checkboxes, that allow users to submit data to a server.

```
<form>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" />
    <button type="submit">Submit</button>
</form>
```

Code 2.27: HTML <form> Tag

Code 2.27 shows an example of the **<form>** tag. This tag is used to create a form in the document, which contains a set of form elements that allow users to submit data to a server.

2.2.1.25 <input />

This tag is used to create an input field in a form. It is a self-closing tag that requires the **type** attribute to specify the type of input field.

```
<form style="display: flex; flex-direction: column; gap: 8px;">
  <div>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" />
  </div>

  <div>
    <label for="age">Age:</label>
    <input type="number" id="age" name="age" min="09" max="100" />
  </div>

  <div>
    <label for="civil-status">Civil Status:</label>
    <input type="radio" id="single" name="civil-status" value="single" />
       Single
    <input type="radio" id="married" name="civil-status" value="married" />
       Married
    <input type="radio" id="divorced" name="civil-status" value="divorced" />
       Divorced
  </div>

  <div>
    <label for="email">Email:</label>
```

```
    <input type="email" id="email" name="email" />
  </div>

  <div>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" />
  </div>

  <div>
    <label for="color">Favorite Color:</label>
    <input type="color" id="color" name="color" />
  </div>

  <div>
    <label for="date">Date of Birth:</label>
    <input type="date" id="date" name="date" />
  </div>

  <div>
    <label for="time">Time of Birth:</label>
    <input type="time" id="time" name="time" />
  </div>

  <div>
    <label for="file">Upload File:</label>
    <input type="file" id="file" name="file" />
  </div>

  <div>
    <label for="message">Message:</label>
    <textarea id="message" name="message"></textarea>
  </div>

  <div>
    <label for="agree">I agree to the terms and conditions:</label>
    <input type="checkbox" id="agree" name="agree" value="yes" />
  </div>

  <button type="submit">Submit</button>
</form>
```

Code 2.28: HTML <input /> Tag

Code 2.28 shows an example of the **<input />** tag. This tag is used to create an input field in a form, which allows users to enter data. The **type** attribute specifies the type of input field, such as text, number, email, or password.

2.2.1.26 <textarea>...</textarea>

This tag is used to create a textarea field in a form. It allows users to enter multiple lines of text.

```
<label for="message">Message:</label>
<textarea id="message" name="message"></textarea>
```

Code 2.29: HTML <textarea> Tag

Code 2.29 shows an example of the **<textarea>** tag. This tag is used to create a textarea field in a form, which allows users to enter multiple lines of text.

2.2.1.27 <button>...</button>

This tag is used to create a button in a form. It is used to submit the form data to a server or perform an action when clicked.

```
<button type="submit">Submit</button>
<button type="reset">Reset</button>
<button type="button">Click Me</button>
```

Code 2.30: HTML <button> Tag

Code 2.30 shows an example of the **<button>** tag. This tag is used to create a button in a form, which allows users to submit the form data to a server or perform an action when clicked. The **type** attribute specifies the type of button, such as submit, reset, or button.

2.2.1.28 <label>...</label>

This tag is used to create a label for an input field in a form. It is used to provide a description or name for the input field. Clicking on the label focuses the associated input field.

```
<label for="name">Name:</label>
<input type="text" id="name" name="name" />
```

Code 2.31: HTML <label> Tag

Code 2.31 shows an example of the **<label>** tag. This tag is used to create a label for an input field in a form, which provides a description or name for the input field. The **for** attribute specifies the ID of the input field that the label is associated with.

2.2.1.29 <select>...</select>

This tag is used to create a dropdown list in a form. It contains a set of **<option>** tags that define the options in the dropdown list.

```
<select id="color" name="color">
    <option value="red">Red</option>
    <option value="green">Green</option>
    <option value="blue">Blue</option>
</select>
```

Code 2.32: HTML <select> Tag

Code 2.32 shows an example of the **<select>** tag. This tag is used to create a dropdown list in a form, which contains a set of **<option>** tags that define the options in the dropdown list. The **id** attribute specifies the ID of the dropdown list, while the **name** attribute specifies the name of the dropdown list.

2.2.1.30 <iframe>...</iframe>

This tag is used to embed another document within the current document. It is used to display content from another website or source. It can also be used to embed videos, maps, or other media.

```
<iframe
  width="600"
  height="450"
  style="border:0;"
  allowfullscreen="true"
  loading="lazy"
  src="https://www.google.com/maps/embed?pb=!1m14!1m12!1m3!1d1075
  .0761255478576!2d123.88209023319297!3d12
  .66699124573315!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13
  .1!5e1!3m2!1sen!2sph!4v1737791388626!5m2!1sen!2sph"
>
</iframe>
```

Code 2.33: HTML <iframe> Tag

Code 2.33 shows an example of the **<iframe>** tag. This tag is used to embed another document within the current document, which displays content from another website or source. The **src** attribute specifies the URL of the document to be embedded, while the **width** and **height** attributes specify the dimensions of the embedded document. In this example, an embedded Google Maps is shown.

2.2.2 More about HTML

You can find more information about HTML tags from the following sources:

- MDN Web Docs HTML Elements
- W3Schools HTML Tags
- Hostinger HTML Cheatsheet

2.3 CSS

CSS (Cascading Style Sheets) is a style sheet that describes how HTML components appear on a page. CSS is used to manage your website's appearance, style, and formatting, including RGB values, border colors, background pictures, and more.

CSS files specify a set of rules for defining a set of properties and their values.

2.3.1 Core Concepts

2.3.1.1 Cascading and Specificity

2.3.1.1.1 Cascading

Cascading refers to the process of combining multiple style sheets and resolving conflicts between them. When multiple style rules apply to the same element, the browser uses a set of rules to determine which styles to apply.

1. **Specificity** - The more specific rule takes precedence.
2. **Order** - The last rule defined takes precedence.
3. **Importance** - The !important rule takes precedence.

In CSS there are three types of style sheets:

```
<p style="color: red;">This is a paragraph.</p>
```

Code 2.34: Inline CSS

1. **Inline Style** - The inline style is defined within the **style** attribute of an HTML element. It is used to define styles for a specific element.

```
<style>
  h1 {
    color: red;
  }
</style>

<h1>This is a heading.</h1>
```

Code 2.35: Internal CSS

2. **Internal Style Sheet** - The internal style sheet is defined within the **<style>** tag in the **<head>** section of the HTML document. It is used to define styles for a specific document.

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

<h1>This is a heading.</h1>
```

Code 2.36: External CSS

3. **External Style Sheet** - The external style sheet is a separate file where you can define all the styles that you want to use on your website. You can link the external style sheet to your HTML document using the **<link>** tag.

2.3.1.1.2 Specificity

Specificity is a set of rules that determines which style rules apply to an element when multiple rules conflict. Specificity is calculated based on the following factors:

1. **Inline Styles** - Inline styles have the highest specificity and override all other styles.
2. **Element Selectors** - Element selectors have the lowest specificity and are overridden by other selectors.
3. **Class Selectors** - Class selectors have a higher specificity than element selectors.
4. **ID Selectors** - ID selectors have a higher specificity than class selectors and element selectors.
5. **!important** - The !important rule overrides all other rules and has the highest specificity.
6. **Order of Appearance** - If two rules have the same specificity, the rule that appears last in the style sheet takes precedence.

2.3.1.2 Selectors

In CSS, selectors are used to target HTML elements and apply styles to them. There are different types of selectors that can be used to target elements based on their type, class, ID, or other attributes.

2.3.1.2.1 Element Selector

The element selector is used to target all elements of a specific type. It is defined by the element name without any additional characters.

```
<p>This is a paragraph.</p>
```

Code 2.37: Element Selector

```
p {
  color: red;
}
```

Code 2.38: Element Selector CSS

Code 2.37 and 2.38 show an example of the element selector. In this example, the **<p>** element is targeted using the element selector, and the color of the text is set to red.

2.3.1.2.2 Class Selector

The class selector is used to target elements with a specific class. It is defined by a period (.) followed by the class name. This selector is useful when you want to apply the same style to multiple elements. This is generally the most common selector used in CSS.

```
<p>This is a <span class="highlight">paragraph</span>.</p>
```

Code 2.39: Class Selector

```
.highlight-text {
  background: yellow;
}
```

Code 2.40: Class Selector CSS

Code 2.39 and 2.40 show an example of the class selector. In this example, the **** element with the class **highlight** is targeted using the class selector, and the background color is set to yellow.

2.3.1.2.3 ID Selector

The ID selector is used to target a specific element with a unique ID. It is defined by a hash (#) followed by the ID name. This selector is used when you want to apply a style to a single element on the page.

```
<p id="intro">This is an introduction.</p>
```

Code 2.41: ID Selector

```
#intro {
  font-size: 24px;
}
```

Code 2.42: ID Selector CSS

Code 2.41 and 2.42 show an example of the ID selector. In this example, the **<p>** element with the ID **intro** is targeted using the ID selector, and the font size is set to 24 pixels.

2.3.1.2.4 Universal Selector

The universal selector is used to target all elements on the page. It is defined by an asterisk (*) character. This selector is used when you want to apply a style to all elements on the page.

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

Code 2.43: Universal Selector CSS

Code 2.43 shows an example of the universal selector. In this example, the universal selector

is used to apply a style to all elements on the page, setting the margin, padding, and box-sizing properties.

2.3.1.2.5 Pseudo-classes

Pseudo-classes are used to define a special state of an element. They are defined by a colon (:) followed by the pseudo-class name. Pseudo- classes are used to style elements based on user interaction or element state.

```
<a href="https://www.github.com/godkingjay">Visit GitHub</a>
```

Code 2.44: Pseudo-class HTML

```
a:link {
  color: blue;
}

a:hover {
  color: red;
}

a:active {
  color: green;
}

a:visited {
  color: purple;
}
```

Code 2.45: Pseudo-class CSS

Code 2.44 and 2.45 show an example of pseudo-classes. In this example, the **<a>** element is targeted using the pseudo-classes **:link**, **:hover**, **:active**, and **:visited**, which define the styles for the link in different states.

2.3.1.2.6 Pseudo-elements

Pseudo-elements are used to style a specific part of an element. They are defined by a double colon (::) followed by the pseudo-element name. Pseudo-elements are used to style elements based on their position or content.

```
<p>Velit sit ad aliquip laborum labore. Excepteur tempor ad duis Lorem. Aute
    labore dolor dolor aliqua eiusmod pariatur ut duis deserunt esse velit.</p>
```

Code 2.46: Pseudo-element HTML

```
p::first-line {
  font-weight: bold;
```

```
}

p::first-letter {
  font-size: 24px;
}

p::before {
  content: "Quote: ";
}

p::after {
  content: " - Author";
}
```

Code 2.47: Pseudo-element CSS

Code 2.46 and 2.47 show an example of pseudo-elements. In this example, the **<p>** element is targeted using the pseudo-elements **::first-line**, **::first-letter**, **::before**, and **::after**, which style the first line, first letter, and add content before and after the paragraph.

2.3.1.3 The Box Model

The box model is a fundamental concept in CSS that defines how elements are displayed on the page. It consists of four parts: content, padding, border, and margin.

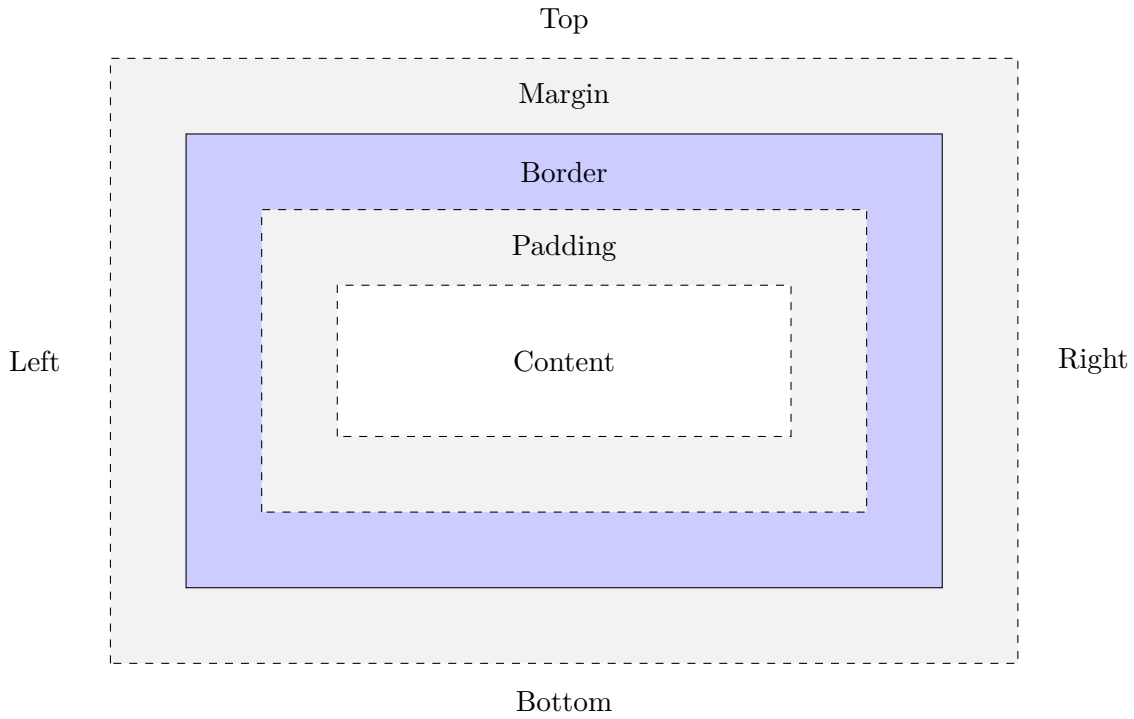

Figure 1: The Box Model

1. **Content** - The content area is where the text and images of the element are displayed. The content area is defined by the width and height of the element.

2. **Padding** - The padding area is the space between the content area and the border of the

element. Padding is used to create space around the content.

3. **Border** - The border area is the line that surrounds the padding and content of the element. The border is used to define the visual appearance of the element.

4. **Margin** - The margin area is the space outside the border of the element. The margin is used to create space between elements.

2.3.1.4 Units

In CSS, there are different units that can be used to define the size of elements. Some of the common units include:

1. **Pixels (px)** - Pixels are a fixed unit of measurement that is used to define the size of elements in terms of screen pixels. Pixels are an absolute unit and do not change based on the screen size.

2. **Percent (%)** - Percentages are a relative unit of measurement that is used to define the size of elements as a percentage of the parent element. Percentages are relative to the parent element and change based on the size of the parent element.

3. **Em (em)** - Em is a relative unit of measurement that is used to define the size of elements relative to the font size of the parent element. Em is relative to the font size of the parent element and changes based on the font size of the parent element.

4. **Rem (rem)** - Rem is a relative unit of measurement that is used to define the size of elements relative to the font size of the root element (html). Rem is relative to the font size of the root element and does not change based on the font size of the parent element.

5. **Viewport Width (vw)** - Viewport width is a relative unit of measurement that is used to define the size of elements relative to the width of the viewport. Viewport width is relative to the width of the viewport and changes based on the size of the viewport.

6. **Viewport Height (vh)** - Viewport height is a relative unit of measurement that is used to define the size of elements relative to the height of the viewport. Viewport height is relative to the height of the viewport and changes based on the size of the viewport.

2.3.2 Properties

2.3.2.1 Colors

In CSS, colors can be specified using different formats, such as color names, hexadecimal values, RGB values, and HSL values.

2.3.2.1.1 Font Color

The **color** property is used to set the color of the text.

```
p {
  color: red;
}
```

Code 2.48: Font Color

Code 2.48 shows an example of the **color** property. In this example, the color of the text in the **<p>** element is set to red.

2.3.2.1.2 Background Color

The **background-color** property is used to set the background color of an element.

```
body {
  background-color: lightblue;
}
```

Code 2.49: Background Color

Code 2.49 shows an example of the **background-color** property. In this example, the background color of the body element is set to light blue.

2.3.2.1.3 Border Color

The **border-color** property is used to set the color of the border of an element.

```
div {
  border: 1px solid black;
}
```

Code 2.50: Border Color

Code 2.50 shows an example of the **border-color** property. In this example, the border color of the ** ** element is set to black.

2.3.2.1.4 Color Names

CSS provides a set of predefined color names that can be used to specify colors. Some of the common color names include:

- **Red** - red
- **Green** - green
- **Blue** - blue
- **Black** - black
- **White** - white
- **Yellow** - yellow
- **Purple** - purple
- **Orange** - orange
- **Gray** - gray
- **Brown** - brown
- **Pink** - pink
- More...

```
p {
  color: blue;
}
```

Code 2.51: Color Names

Code 2.51 shows an example of using color names to set the color of the text in the **<p>** element to blue.

2.3.2.1.5 Hexadecimal Colors

Hexadecimal colors are represented using a six-digit code that defines the amount of red, green, and blue as well as the transparency of the color. The code starts with a hash (#) followed by the six-digit hexadecimal code - **#RRGGBB**. Eight digits can be used to include the alpha channel - **#RRGGBBAA**.

```
p {
  color: #ff0000; /* Red */
  background-color: #00ff00; /* Green */
  border-color: #0000ff; /* Blue */
}

div {
  background: #ffff00ff; /* Yellow with 100% opacity */
}
```

Code 2.52: Hexadecimal Colors

Code 2.52 shows an example of using hexadecimal colors to set the color of the text, background, and border of the **<p>** element to red, green, and blue, respectively. The color of the text in the ** ** element is set to yellow with 100% opacity.

2.3.2.1.6 RGB or RGBA Colors

RGB colors are represented using the RGB color model, which defines the amount of red, green, and blue in the color. The RGB color is defined using the **rgb()** function with three values for red, green, and blue. The **RGBA colors** include an additional value for the alpha channel (transparency) and are defined using the **rgba()** function.

```
p {
  color: rgb(255, 0, 0); /* Red */
  background-color: rgba(0, 255, 0, 0.5); /* Green with 50% opacity */
}
```

Code 2.53: RGB Colors

Code 2.53 shows an example of using RGB and RGBA colors to set the color of the text in the **<p>** element to red and the background color to green with 50% opacity.

2.3.2.1.7 HSL or HSLA Colors

HSL colors are represented using the HSL color model, which defines the hue, saturation, and lightness of the color. The HSL color is defined using the **hsl()** function with three values

for hue, saturation, and lightness. The **HSLA colors** include an additional value for the alpha channel (transparency) and are defined using the **hsla()** function.

```
p {
  color: hsl(0, 100\%, 50\%); /* Red */
  background-color: hsla(120, 100\%, 50\%, 0.5); /* Green with 50\% opacity */
}
```

Code 2.54: HSL Colors

Code 2.54 shows an example of using HSL and HSLA colors to set the color of the text in the **<p>** element to red and the background color to green with 50% opacity.

2.3.2.1.8 Gradients

Gradients are used to create a smooth transition between two or more colors. Gradients can be linear or radial and can be defined using the **linear-gradient()** or **radial-gradient()** function.

```
p {
  background: linear-gradient(to right, red, blue);
}
```

Code 2.55: Gradients

Code 2.55 shows an example of using gradients to create a linear gradient background for the **<p>** element that transitions from red to blue.

2.3.2.1.9 CSS Background Image

The **background-image** property is used to specify the background image for an element. The value can be a URL to an image file or a gradient.

```
div {
  background-image: url('https://picsum.photos/200');
}
```

Code 2.56: Background Image

Code 2.56 shows an example of using the **background-image** property to set the background image for the ** ** element to an image from the URL.

2.3.2.1.10 CSS Background Size

The **background-size** property is used to specify the size of the background image. It can be set to a specific size, such as **cover** or **contain**, or to a percentage of the element's width and height.

```
div {
```

```
  background-image: url('https://picsum.photos/200');
  background-size: cover;
}
```

Code 2.57: Background Size

Code 2.57 shows an example of using the **background-size** property to set the size of the background image for the ** ** element to cover the entire element.

- MDN Web Docs CSS Reference
- W3Schools CSS Reference
- Toptal CSS Cheat Sheet

3

Version Control

4

NextJS

5

Mobile Applications Development

6

Cloud Computing

7

Artificial Intelligence

8

Internet of Things and Augmented Reality

A. Books

-

B. Other Sources

-