30 Days Front-End Development + Alaybee Sports Project Plan

Day 1 – Introduction to HTML

1 . HTML is the structure of every web page.

2 . It uses tags to define elements.

3 . The file extension is `.html`.

4 . `<head>` contains metadata, `<body>` has visible content.

5 . Always start with `<!DOCTYPE html>`.

Example:

html

```
<!DOCTYPE html>
<html>
<head><title>Intro</title></head>
<body><h1>Welcome to HTML</h1></body>
</html>
```

Day 2 – Text Formatting

1 . Text formatting makes content readable.

2 . `<b>`, `<i>`, `<u>` add emphasis.

3 . `<sup>`, `<sub>` are used in math/science text.

4 . `<mark>` highlights text.

5 . Formatting tags are inline elements.

Example:

html

```
<b>Bold</b> <i>Italic</i> <u>Underline</u>
```

Day 3 – Lists

1 . Lists organize data clearly.

2 . `<ul>` creates unordered lists.

3 . `<ol>` creates ordered lists.

4 . `<li>` represents list items.

5 . Lists are used for menus and content grouping.

Example:

html

```
<ul><li>Apple</li><li>Banana</li></ul>
```

Day 4 – Links and Images

1 . `<a>` tag is used for hyperlinks.

2 . `href` specifies destination URL.

3 . `<img>` displays an image on a page.

4 . Use `alt` for alternative text.

5 . `target="_blank"` opens new tab.

Example:

html

```
<a href="https://anvistar.in" target="_blank">Visit Anvistar</a>
<img src="logo.png" width="150">
```

Day 5 – Tables

1 . Tables display structured data.

2 . `<tr>` defines a row, `<td>` defines a cell.

3 . `<th>` is for headers.

4 . You can merge cells using `colspan`.

5 . Add `border` for outline.

Example:

html

```
<table border="1">
<tr><th>Name</th><th>Course</th></tr>
<tr><td>Amit</td><td>Java</td></tr>
</table>
```

Day 6 – Forms

1 . Forms collect user input.

2 . `<input>` takes text, email, password.

3 . `<select>` creates dropdowns.

4 . `<textarea>` is used for large text.

5 . `<button>` submits form data.

Example:

html

```
<form>
 <input type="text" placeholder="Name">
 <input type="email" placeholder="Email">
</form>
```

Day 7 – HTML5 Semantic Tags

1 . Semantic tags describe meaning.

2 . `<header>`, `<footer>`, `<article>` improve clarity.

3 . Better for SEO and accessibility.

4 . Replace generic `<div>` with semantic ones.

5 . Makes code clean and professional.

Example:

html

```
<header><h2>Welcome</h2></header>
<footer>© 2025</footer>
```

Day 8 – Media Tags

1 . HTML5 supports audio and video tags.

2 . `<video>` and `<audio>` embed media files.

3 . `controls` attribute adds buttons.

4 . Use `<source>` for different file types.

5 . Supports MP4, MP3 formats.

Example:

html

```
<video controls width="200">
 <source src="intro.mp4" type="video/mp4">
</video>
```

Day 9 – Introduction to CSS

1 . CSS adds design and layout to web pages.

2 . You can use inline, internal, or external CSS.

3 . The syntax is `selector { property:value; }`.

4 . Styles make websites attractive.

5 . CSS separates content from design.

Example:

html

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

Day 10 – CSS Selectors

1 . Selectors target specific HTML elements.

2 . ID uses `#idname`.

3 . Class uses `.classname`.

4 . Group selectors style multiple tags.

5 . Universal selector is ` `.

Example:

css

```
#title { color:red; }

.text { font-size:18px; }
```

Day 11 – CSS Box Model

1 . Box model manages layout spacing.

2 . Includes margin, border, padding, and content.

3 . Each element behaves like a box.

4 . Helps control alignment and design.

5 . Important for responsive layout.

Example:

css

```
div { border:1px solid black; padding:10px; margin:10px; }
```

Day 12 – CSS Positioning

1 . Position defines how elements are placed.

2 . Values: static, relative, absolute, fixed.

3 . `top`, `left`, `right`, `bottom` define placement.

4 . Useful for overlays or menus.

5 . Z-index controls overlapping order.

Example:

css

```
.box { position:absolute; top:20px; left:30px; }
```

Day 13 – CSS

1 . Flexbox arranges elements in a flexible row or column.

2 . Use `display:flex;` for container.

3 . Align items with `justify-content` and `align-items`.

4 . Helps in responsive layouts.

5 . Easy to center content.

Example:

css

```
.container { display:flex; justify-content:center; }
```

Day 14 – CSS Grid

1 . CSS Grid manages two-dimensional layouts.

2 . Use `display:grid;` to activate.

3 . Define rows and columns.

4 . Adjust gaps using `grid-gap`.

5 . Ideal for dashboards.

Example:

css

```
.grid { display:grid; grid-template-columns:1fr 1fr; gap:10px; }
```

Day 15 – CSS Transitions

1 . Transitions create smooth effects.

2 . Use `transition` property on hover.

3 . Combine with background or color change.

4 . Duration defines speed.

5 . Makes UI more engaging.

Example:

css

```
div:hover { background:yellow; transition:0.5s; }
```

Day 16 – JavaScript Basics

1 . JavaScript adds behavior to web pages.

2 . Variables store data.

3 . Statements end with semicolons.

4 . Case-sensitive language.

5 . Output via `alert()` or `document.write()`.

Example:

html

```
<script>
alert("Welcome to JavaScript");
</script>
```

Day 17 – Conditions and Loops

1 . `if`, `else` control logic.

2 . Loops repeat code automatically.

3 . `for`, `while`, `do-while` are main loops.

4 . Use braces `{}` for multiple lines.

5 . Useful for calculations.

Example:

html

```
<script>
for(let i=1;i<=3;i++){ document.write(i); }
```

</script>

Day 18 – Functions

1 . Functions are reusable code blocks.

2 . Use `function name(){}` syntax.

3 . Can accept parameters.

4 . Return values using `return`.

5 . Keep code modular.

Example:

html

```
<script>
function greet(){ alert("Hello Amit"); }
greet();
</script>
```

Day 19 – Arrays and Objects

1 . Arrays store multiple values.

2 . Objects store key-value pairs.

3 . Access array elements by index.

4 . Access object values by keys.

5 . Useful for managing data.

Example:

html

```
<script>
let student={name:"Amit",course:"JS"};
document.write(student.name);
</script>
```

Day 20 – DOM Manipulation

1 . DOM controls webpage content dynamically.

2 . Use `document.getElementById()` to select.

3 . Change text or HTML using `.innerText`.

4 . Respond to events using JS.

5 . Basis of interactive pages.

Example:

html

```
<p id="msg"></p>
<script>
document.getElementById("msg").innerText="Welcome!";
</script>
```

Project Work: Alaybee Sports Website

Day 21 – Project Setup

Create folder `AlaybeeSports`.

Add `index.html`, `style.css`, `script.js`.

Connect all files.

Test basic layout.

Add Bootstrap CDN.

Day 22 – Home Page Layout

Create Header and Footer using Bootstrap.

Add navigation bar with logo.

Add hero banner section.

Use background image for sports theme.

Make layout responsive.

Day 23 – Slider Section

Use Bootstrap carousel.

Add 3–4 sports images.

Include captions for each slide.

Enable automatic slide change.

Style using CSS for smooth look.

Day 24 – About Us Section

Create a section describing company vision.

Add sports-related text and images.

Use columns for clean layout.

Add “Read More” button.

Make it mobile responsive.

Day 25 – Services Section

List main services (Training, Equipment, Coaching).

Use cards or icons.

Add hover effects.

Write short descriptions.

Align using Bootstrap grid.

Day 26 – Products Section

Create 3x3 product grid.

Each card has image, name, price.

Add “Buy Now” button.

Use shadow effect for style.

Add responsiveness.

Day 27 – Contact Us Form

Create form with name, email, subject, message.

Add Bootstrap input styling.

Validate fields with JS.

Submit button with hover effect.

Show success message after submit.

Day 28 – Footer Section

Include address, contact info, social links.

Use icons from FontAwesome.

Add copyright line.

Align center with background color.

Responsive design.

Day 29 – Complete Integration

Combine all sections into `index.html`.

Check all links, forms, and responsiveness.

Apply color theme.

Test on mobile and desktop.

Fix layout issues.

Day 30 – Final Presentation

Review all code structure.

Prepare final documentation.

Test site in Chrome and Edge.

Take screenshots for report.