**FRONT-END AND BACK-END LANGUAGES**

To build an interactive website, you'll need to use a combination of front-end and back-end languages. Here’s a breakdown:

### Front-End Languages

These are used to create the user interface and experience.

1. \*\*HTML (HyperText Markup Language)\*\*

- Structure of the web pages.

2. \*\*CSS (Cascading Style Sheets)\*\*

- Styling and layout of the web pages.

3. \*\*JavaScript\*\*

- Adds interactivity, handles events, and manipulates the DOM.

- Popular libraries/frameworks:

- \*\*jQuery\*\* (simplifies JavaScript)

- \*\*React\*\* (component-based UI)

- \*\*Vue.js\*\* (progressive framework)

- \*\*Angular\*\* (full-fledged framework)

4. \*\*TypeScript\*\*

- A superset of JavaScript that adds static typing, useful for larger projects.

### Back-End Languages

These handle the server-side logic, database interactions, and application functionality.

1. \*\*Node.js (JavaScript runtime)\*\*

- Allows you to use JavaScript on the server side.

2. \*\*Python\*\*

- Popular frameworks:

- \*\*Django\*\* (high-level framework)

- \*\*Flask\*\* (micro-framework)

3. \*\*Ruby\*\*

- \*\*Ruby on Rails\*\* is a popular framework for building web applications.

4. \*\*PHP\*\*

- Widely used for server-side scripting, often with frameworks like Laravel or Symfony.

5. \*\*Java\*\*

- Commonly used with frameworks like Spring.

6. \*\*C#\*\*

- Often used with ASP.NET for building web applications.

### Databases

You’ll also need a way to store and manage data.

1. \*\*SQL Databases\*\*

- \*\*MySQL\*\*

- \*\*PostgreSQL\*\*

- \*\*SQLite\*\*

2. \*\*NoSQL Databases\*\*

- \*\*MongoDB\*\*

- \*\*Firebase\*\* (also offers real-time database capabilities)

### Additional Tools and Technologies

- \*\*Version Control:\*\* Git for collaboration and version management.

- \*\*APIs:\*\* RESTful services or GraphQL for communication between front-end and back-end.

- \*\*Web Servers:\*\* Nginx or Apache for hosting your applications.

- \*\*Frameworks:\*\* Use frameworks like Next.js (for React) or Nuxt.js (for Vue) for server-side rendering and static site generation.

Choosing the right languages and frameworks depends on your project requirements, team skills, and personal preferences. Let me know if you need more details on any specific technology!

**VIDEO TAG IN HTML**

In HTML, the `<video>` tag is used to embed video content on a web page. It comes with several attributes that control its behavior and appearance. Here's a breakdown of the `<video>` tag and its control properties:

### Basic Structure

```html

<video width="640" height="360" controls>

<source src="video.mp4" type="video/mp4">

<source src="video.ogg" type="video/ogg">

Your browser does not support the video tag.

</video>

```

### Key Attributes

1. \*\*`src`\*\*: Specifies the URL of the video file.

2. \*\*`controls`\*\*: Adds basic controls like play, pause, and volume to the video player.

3. \*\*`autoplay`\*\*: If present, the video will start playing automatically when the page loads.

4. \*\*`loop`\*\*: If present, the video will loop continuously after it finishes playing.

5. \*\*`muted`\*\*: If present, the video will start muted.

6. \*\*`poster`\*\*: Specifies an image to be shown while the video is downloading, or until the user hits the play button.

7. \*\*`preload`\*\*: Specifies if and how the author thinks the video should be loaded when the page loads:

- `none`: Do not load the video.

- `metadata`: Load only metadata (dimensions, duration).

- `auto`: Load the video when the page loads.

8. \*\*`playsinline`\*\*: Allows the video to play inline on mobile devices rather than going fullscreen.

### Example Usage

```html

<video width="640" height="360" controls autoplay muted loop poster="thumbnail.jpg">

<source src="video.mp4" type="video/mp4">

<source src="video.ogg" type="video/ogg">

Your browser does not support the video tag.

</video>

```

### Additional Properties

You can also use JavaScript to control the video playback programmatically:

- \*\*`play()`\*\*: Starts playback.

- \*\*`pause()`\*\*: Pauses the video.

- \*\*`currentTime`\*\*: Gets or sets the current playback position (in seconds).

- \*\*`duration`\*\*: Returns the total duration of the video (in seconds).

- \*\*`volume`\*\*: Gets or sets the volume level (0.0 to 1.0).

- \*\*`muted`\*\*: Gets or sets the muted state of the video.

### Example JavaScript Control

```html

<script>

const video = document.querySelector('video');

// Play the video

function playVideo() {

video.play();

}

// Pause the video

function pauseVideo() {

video.pause();

}

// Set volume

function setVolume(vol) {

video.volume = vol;

}

</script>

```

### Conclusion

The `<video>` tag is a powerful tool for embedding videos in HTML, providing a range of attributes and properties to customize the user experience.