SPPU MCA Sem 1 - Web Tech (HTML5) - Exam Notes

1. Client-Server Architecture

Concept: Foundation of the web. The *client* (browser) requests data, the *server* processes the request and sends a response.

Why Important?

- **Scalability:** Can handle more users by adding servers.
- **Security:** Centralized data protection.
- **Manageability:** Easy to update and maintain server logic.

Simple Code Analogy:

```html
<!-- This is the Client-Side (Frontend) -->
<form action="/login" method="POST"> <!-- Request to Server -->
  <input type="text" name="username">
  <input type="password" name="password">
  <button type="submit">Login</button> <!-- User action sends request -->
</form>
```

The server has code (e.g., in Node.js/Python) to check the `username` and `password` and send a response like "Login Success" or "Failed".

---

2. Frontend vs. Backend

Aspect	**Frontend (Client-Side)**	**Backend (Server-Side)**
Languages	HTML, CSS, JavaScript	Node.js, Python, PHP, Java, C#
Database	No direct access	MySQL (SQL), MongoDB (NoSQL)
Runs On	User's browser	Remote server
Purpose	What user **sees** and **interacts** with.	**Logic**, **data processing**, **security**.

---

3. HTML5 Semantic Elements

Concept: Tags that give **meaning** to content. Better for **SEO** and **Accessibility**.

Why use? `<div class="header">` vs. `<header>` - the second one is instantly understood by browsers and developers.

Key Tags & Code:

html

```html
<!DOCTYPE html>
<html>
<body>
  <header>
    <h1>My Website</h1>
    <nav>
      <a href="/home">Home</a> | <a href="/about">About</a>
    </nav>
  </header>

  <main>
    <article>
      <h2>Blog Post Title</h2>
      <p>This is an independent article.</p>
    </article>
    <section>
      <h3>Comments Section</h3>
      <p>User comments go here.</p>
    </section>
  </main>

  <footer>
    <p>&copy; 2025 My Site</p>
  </footer>
</body>
</html>
```

---

4. HTML5 `<video>` & `<audio>` Tags

Purpose: Embed media without plugins like Flash.

Basic Code (Must Know):

html

```html
<!-- VIDEO TAG -->
<video width="320" height="240" controls poster="preview.jpg">
  <source src="movie.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>

<!-- AUDIO TAG -->
<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>
```

Key Attributes:

- `controls`: Shows play/pause/volume.
- `autoplay`: Starts automatically (often needs `muted`).
- `muted`: Starts without sound.
- `loop`: Repeats infinitely.
- `poster` (video only): Preview image.

---

5. HTML5 Graphics: Canvas vs. SVG

Feature	Canvas	SVG (Scalable Vector Graphics)
Type	Raster (Pixel-based)	Vector (Shape-based)

Code Example

html

```html
<canvas id="myCanvas" width="200" height="100"></canvas>
<script>
  var ctx = document.getElementById('myCanvas').getContext('2d');
  ctx.fillStyle = 'red';
```

```
    ctx.fillRect(10, 10, 150, 75); // Draws a red rectangle
</script>
```

|

html

```html
<svg width="200" height="100">
    <rect x="10" y="10" width="150" height="75" fill="red"/>
</svg>
```

|
| **Best For** | Games, dynamic charts, photo manipulation | Logos, icons, maps, diagrams |
| **Scalability** | Loses quality when zoomed | Perfect quality at any zoom level |

---

6. HTML5 APIs (Application Programming Interfaces)

1. Web Storage API (`localStorage`)

- Stores data in the browser **permanently** (until cleared).

html

```html
<script>
    // SAVE data
    localStorage.setItem("username", "JohnDoe");

    // GET data
    let user = localStorage.getItem("username");
    alert("Welcome " + user); // Alerts "Welcome JohnDoe"

    // REMOVE data
    // localStorage.removeItem("username");
</script>
```

2. Geolocation API

- Gets the user's geographical location.

html

```html
<script>
    function getLocation() {
```

```
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(showPosition);
    }
  }
  function showPosition(position) {
    alert("Latitude: " + position.coords.latitude + ", Longitude: " + position.coords.longitude);
  }
</script>
<button onclick="getLocation()">Get Location</button>
```

3. Drag and Drop API

- Makes any element draggable.

html

```
<!DOCTYPE html>
<html>
<style>
  #dropZone { width: 200px; height: 100px; border: 2px dashed #ccc; }
</style>
<body>
  <!-- Draggable Element →
  <div draggable="true" id="dragItem">Drag me!</div>

  <!-- Drop Zone →
  <div id="dropZone" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

  <script>
    function allowDrop(event) { event.preventDefault(); }
    function drag(event) { event.dataTransfer.setData("text", event.target.id); }
    function drop(event) {
      event.preventDefault();
      var data = event.dataTransfer.getData("text");
      event.target.appendChild(document.getElementById(data));
    }
  </script>
</body>
</html>
```

Key Events: `dragstart`, `dragover`, `drop`.

---

7. Git & GitHub - Version Control

Core Concept: Tracks changes in your code. Essential for collaboration.

Basic Workflow & Commands:

```bash
# 1. Initialize a new Git repository in your project folder
git init

# 2. Check the status of your files
git status

# 3. Add all files to the "staging area" (prepare to save)
git add .

# 4. Permanently save (commit) the changes with a message
git commit -m "My first commit, added login page"

# 5. Link your local repo to a remote one on GitHub
git remote add origin https://github.com/yourname/yourrepo.git

# 6. Upload (push) your code to GitHub
git push -u origin main
```

Key Terms:

- **Repository (Repo):** Your project folder tracked by Git.
- **Commit:** A saved snapshot of your changes.
- **Push:** Uploading commits to a remote server (e.g., GitHub).
- **Pull:** Downloading the latest changes from the remote server.

Final Exam Tips:

1. **For 5-Mark Questions:** Be ready to write a short code snippet (e.g., for `<video>` tag or `localStorage`).
2. **For 10-Mark Questions:** Prepare to compare technologies (e.g., Canvas vs. SVG, SQL vs. NoSQL) using a table and write 1-2 paragraphs.

3. **Always mention "Why":** Why use semantic tags? (SEO, Accessibility). Why use Client-Server? (Scalability, Security).
4. **Draw Diagrams:** For Client-Server architecture, a simple diagram with labels earns good marks.

Master these concepts and code examples. They form the bedrock of modern web development and are crucial for your success in the exam and your future career.

Good luck.