Beginner’s Guide to JavaScript DOM Manipulation Projects

Subtitle: Learn to build interactive web apps by dynamically updating HTML elements using JavaScript.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

The **DOM (Document Object Model)** allows JavaScript to interact with web pages dynamically. Understanding DOM manipulation is essential for creating interactive apps, updating content, and responding to user actions in real-time.

Step 1: Accessing DOM Elements

- Select elements using JavaScript:

```
const header = document.getElementById('header');
const buttons = document.querySelectorAll('.btn');
```

- **Tip:** `querySelector` returns the first match, `querySelectorAll` returns all matches

---

Step 2: Changing Content & Attributes

```
header.textContent = "Welcome to My App";
const link = document.querySelector('a');
link.setAttribute('href', 'https://dev-roast-app.vercel.app');
```

- Update styles dynamically:

```
header.style.color = 'blue';
header.style.fontSize = '24px';
```

Step 3: Creating & Removing Elements

- Create a new element:

```
```

```
const newDiv = document.createElement('div');
newDiv.textContent = "Hello World";
document.body.appendChild(newDiv);
```

- Remove an element:

```
newDiv.remove();
```

Step 4: Handling Events

- Add event listeners to elements:

```
const button = document.querySelector('.btn');
button.addEventListener('click', () => alert('Button clicked!'));
```

- Common events: `click`, `input`, `mouseover`, `keydown`

Step 5: Project 1 – Interactive To-Do List

- Add tasks dynamically
- Mark tasks as complete
- Remove tasks
- Update the DOM based on user actions

Step 6: Project 2 – Dynamic Theme Toggle

- Add a button to switch between light/dark mode
- Update CSS classes dynamically
- Store user preference in Local Storage for persistence

Step 7: Project 3 – Live Search Filter

- Display a list of items
- Use `input` event to filter items as the user types

```
const searchInput = document.getElementById('search');
searchInput.addEventListener('input', () => {
```

```
  const filter = searchInput.value.toLowerCase();
  items.forEach(item => {
    item.style.display = item.textContent.toLowerCase().includes(filter) ? 'bloc
  });
});
```

Step 8: Best Practices

- Use meaningful element IDs and class names
- Avoid inline JavaScript; use event listeners
- Update the DOM efficiently (e.g., batch changes when possible)
- Always validate user input before manipulating DOM

Step 9: Key Takeaways

- DOM manipulation makes web pages interactive
- Learn to select, update, create, and remove elements dynamically
- Combine with events for responsive apps
- Practice with small projects to solidify skills

Visit **haas.dev** for step-by-step DOM manipulation projects, interactive tutorials, and beginner-friendly JavaScript resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
