JavaScript Cheat Sheet: DOM Events & Asynchronous Programming

Page 1: What is the DOM?

The DOM (Document Object Model) is how JavaScript 'sees' your webpage.

When you write HTML, the browser turns it into a tree-like structure (the DOM), and JavaScript can interact with it.

Examples:

- Change text on the page
- Respond to clicks
- Show or hide elements

Basic Code:

```
document.getElementById("myButton").innerText = "Clicked!";
```

This line finds the button with ID 'myButton' and changes its text.

JavaScript Cheat Sheet: DOM Events & Asynchronous Programming

Page 2: Introduction to DOM Events

DOM events let you respond to user actions like clicks, keypresses, or page loads.

Common Events:

- click: when a user clicks something
- input: when user types in a box
- submit: when a form is submitted

Example:

```
document.getElementById("myButton").addEventListener("click", function() {
  alert("Button clicked!");
});
```

This waits for a click, then runs the function (which shows a message).

JavaScript Cheat Sheet: DOM Events & Asynchronous Programming

Page 3: Writing Better Event Handlers

Instead of writing code inline (bad practice), use separate functions.

Example:

```
function handleClick() {
  alert("Clicked!");
}

const btn = document.getElementById("myButton");
btn.addEventListener("click", handleClick);
```

Why this is better:

- Clean and reusable
- Easier to debug
- Separation of logic from layout

JavaScript Cheat Sheet: DOM Events & Asynchronous Programming

Page 4: What is Asynchronous JavaScript?

JavaScript normally runs line by line. But some tasks (like fetching data from a server) take time.

Async means: "Don't wait for it, keep going, and come back when ready."

Example:

```
console.log("Start");
setTimeout(() => {
  console.log("Later");
}, 1000);
console.log("End");
```

Output:

Start

End

Later

The 'Later' line happens after 1 second, while the rest keeps going.

JavaScript Cheat Sheet: DOM Events & Asynchronous Programming

Page 5: Using fetch and async/await

You can use fetch to get data from a server without reloading the page.

Example with Promises:

```
fetch("https://api.example.com/data")
  .then(response => response.json())
  .then(data => {
    console.log(data);
  });
```

Same with async/await (simpler):

```
async function loadData() {
  const response = await fetch("https://api.example.com/data");
  const data = await response.json();
  console.log(data);
}

loadData();
```