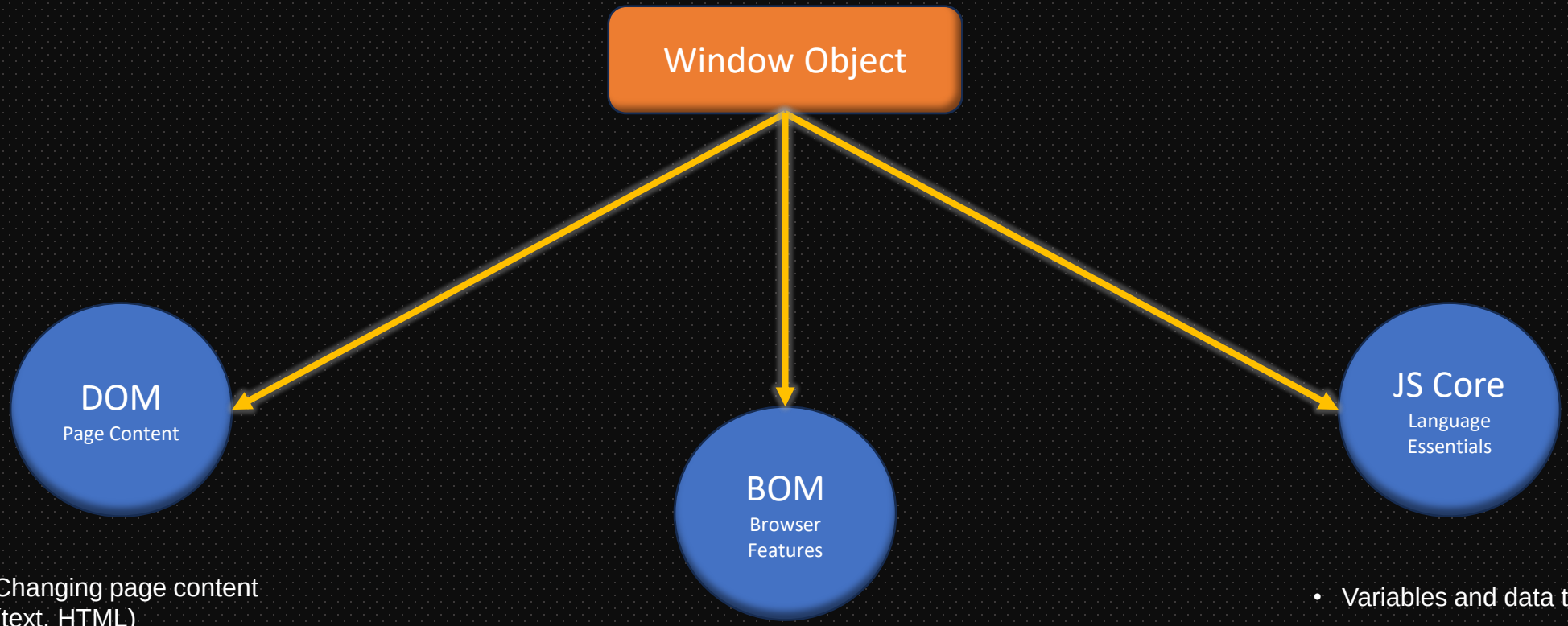


- Changing page content (text, HTML)
- Changing element styles (colors, fonts, sizes)
- Handling user events (clicks, typing, scrolling)

- Alerts
- Confirmations
- Prompts
- Browser window size
- URL, history navigation
- Timers

- Variables and data types
- Operators
- Functions
- Objects and arrays

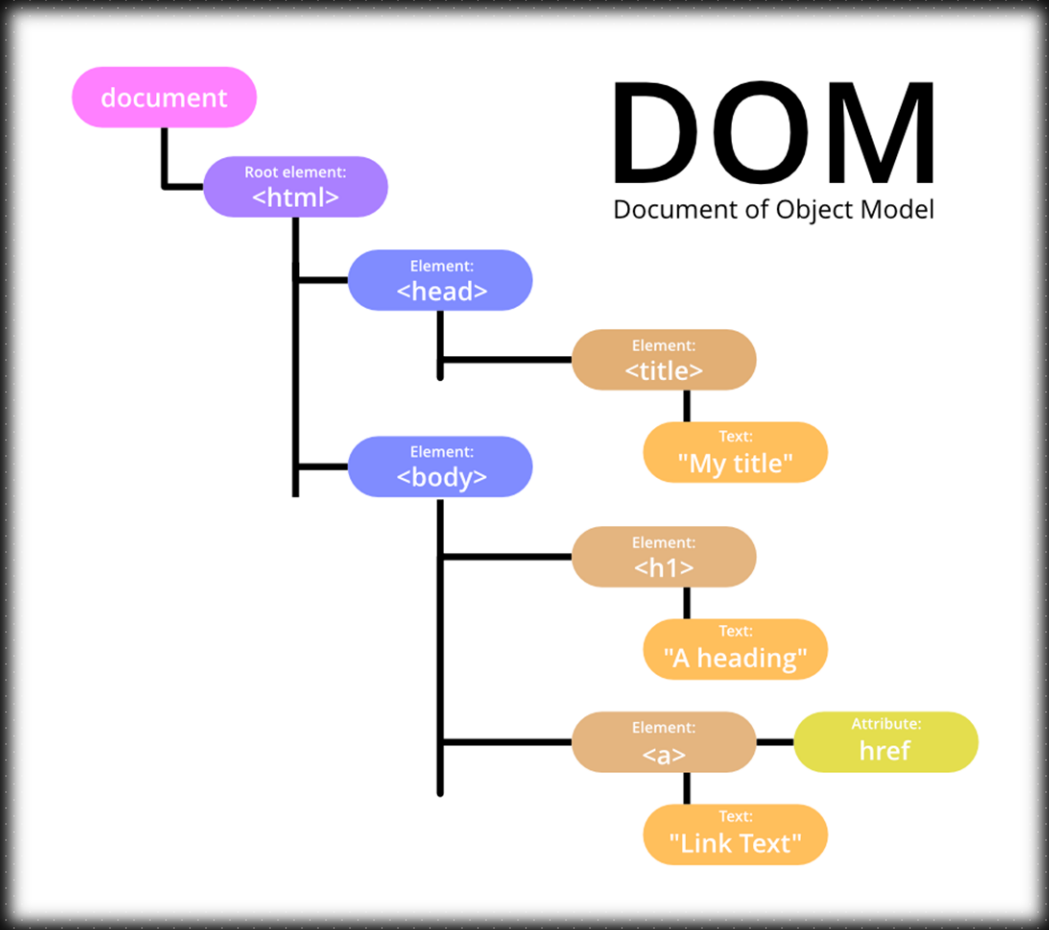
5 Phase Or Pillars Of Dom Manipulation:

1. DOM (Document Object Model)
2. Selection Of HTML
3. HTML Manipulation (Changin the HTML)
4. CSS Manipulation (Changin the CSS)
5. Event Listeners (Event Handling)

DOM
(Document Object Model)

What is DOM?

When a web page is loaded, browser creates a document object model (DOM) of the page.



Let's take a deep dive:

- The HTML page gets converted into the DOM (a big JavaScript-like object structure named “Document”).
- Because of that, everything on the page — buttons, headings, images — becomes an object with properties and methods.
- That's why you can do things like:
  - querySelector("h1") ➔ to find an element.
  - addEventListener("click", ...) ➔ to listen for user actions.
  - innerHTML = “Mai hun manas" ➔ to change content.

Note1:

- Document object is available inside window object.

Note2:

- If your <script> is in the <head>, the browser runs the JavaScript before it finishes loading the <body>. In that case, document Elements you are trying to access are null.

Without the DOM, JavaScript wouldn't know what’s on the page, and we couldn’t control it so easily.

In short:

HTML ➔ DOM ➔ JavaScript controls the DOM like objects.

What Is NodeList?

A NodeList is a simple list of DOM elements or collection of nodes (DOM elements) you got from the webpage, but it’s not a full real array.

Feature	NodeList	Array
Definition	Collection of nodes (DOM elements)	List of any data types (numbers, strings, objects, etc.)
Creation	Returned by DOM methods like querySelectorAll()	Created with [] or Array()
Type	object (specifically a NodeList)	object (specifically an Array)
Indexable	Yes (like nodelist[0])	Yes (like array[0])
Length Property	Yes	Yes
Loopable	Yes (with for, sometimes forEach)	Yes (with for, forEach, etc.)
Array Methods	No (older NodeLists) or **some** (modern NodeLists have forEach)	Yes (full support: map, filter, reduce, etc.)
Static/Dynamic	Usually static (doesn’t auto-update if DOM changes)	Static (unless manually updated)
Convert to Array	Needed if you want full Array methods (Array.from(nodelist) or [...nodelist])	Already an Array

Selection

Selection:

Selecting with tag:

```
document.getElementsByTagName("h1")
```

Selecting with id:

```
document.getElementById("id")
```

Selecting with class:

```
document.getElementsByClassName("class")
```

Query selector:

```
document.querySelector("id/class/tag")
```

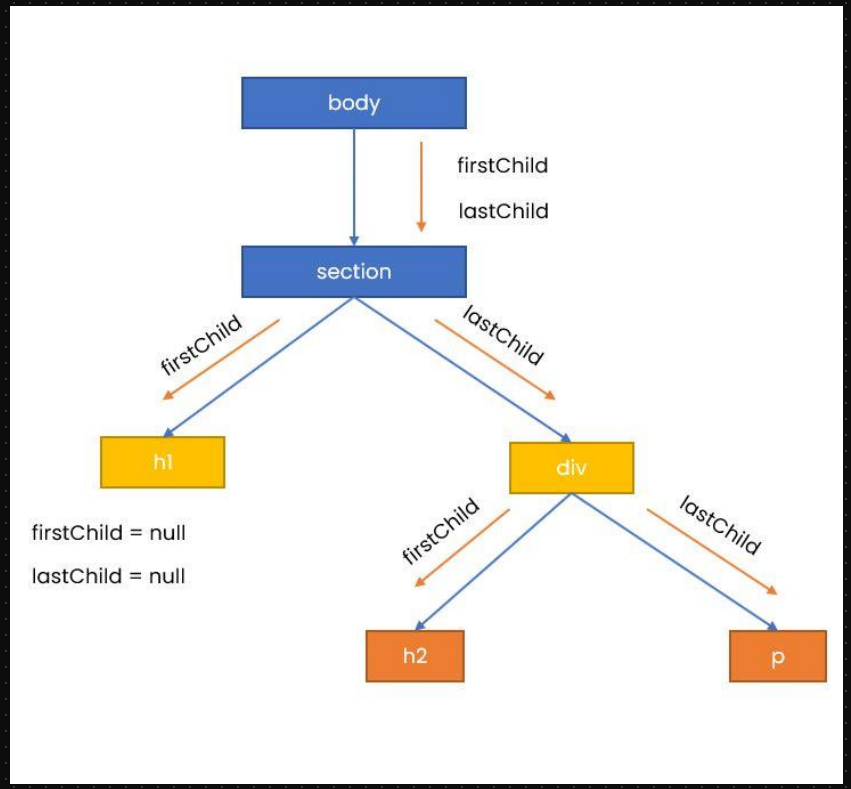
It returns only first matched element.

```
document.querySelectorAll("id/class/tag")
```

It returns all matched elements in the form of NodeList.

Note:

- For id use '#'
- For class use '.'



Do You Know About $0 Magic

Manipulating The HTML

Changing the HTML (Manipulating the HTML):

Properties:

tagName (read only): Returns the tag name (like "DIV", "H1") of the element in UPPERCASE. (include hidden elements)

nodeName (read only): Returns the name of the node in UPPERCASE for element nodes (e.g., "DIV", "SPAN") and special strings for other types (e.g., "#text" for text nodes, "#comment" for comment nodes).(includes hidden elements and works on any node type, not just HTML elements)

innerText: Returns all the text of the element and its children. (It respects styles like display: none or visibility: hidden, so hidden elements are ignored.)

innerHTML: Returns the HTML content (including any tags inside) as a string. (include hidden elements)

textContent: Returns all the text of the element and its children (include hidden elements).

Property	Includes hidden elements?	Includes HTML tags?
innerText	❌ No (ignores hidden text)	❌ No
innerHTML	✅ Yes	✅ Yes (returns tags too)
textContent	✅ Yes	❌ No

Note:

- ❑ We can also set the html using these properties

Insert Elements (Addition Of Elements):

Step1: Creation of element

```
let elem = document.createElement('div');
```

Step2: Creation of element

node.append(elem) : adds at the end of node (inside)

node.prepend(elem) : adds at the start of node (inside)

node.after(elem) : adds after the node (outside)

node.before(elem) : adds before the node (outside)

node.inserAdjacentElement(position, elem) : positon can be “beforebegin”, “afterbegin”, “beforeend”, “afterend”

Deletion of Elements:

node.remove()

Parent Node

node.parentNode: Returns the immediate parent of a node (could be an Element, Document, or DocumentFragment).

Common Parent Methods

replaceChild(newChild, oldChild): Replaces an existing child node with a new one.

appendChild(child): Adds a child node to the end of the parent's children list.

insertBefore(newNode, referenceNode): Inserts a new node before a specified existing child node.

removeChild(child): Removes a specified child node from the parent.

Note:

- sometimes you see appendChild and removeChild in older code. (only work with nodes and not with strings like text).

Attributes:

getAttribute(attr): To get the attribute value

setAttribute(attr, value): To set the attribute value

Example:

```
let node = document.querySelector('#btn');

let beforeStyle = node.getAttribute('style');
console.log(beforeStyle); // color: blue;

node.setAttribute('style', 'background-color: red');

let afterStyle = node.getAttribute('style');
console.log(afterStyle); // background-color: red;
```

1. Create a paragraph with text “mai tumse pyar nahi karta hun” and add background color to black and font color to green using javascript.

2. Insert a button with text “click me” as the first element inside the paragraph created in 1st question.

3. Create a tag in html and give it a class & some styling. Now create a new class in css and try to append this class to the element.

Challenge 1:

Create a function that takes node and newTagName and replace the node from the new node with changed tag name in the DOM.

```
function replaceTag(node, newTagName) {
    if (!node || !(node instanceof Element)) {
        console.error('Invalid node provided');
        return null;
    }

    const newNode = document.createElement(newTagName);

    // Copy attributes
    for (const attr of node.attributes) {
        newNode.setAttribute(attr.name, attr.value);
    }

    // Copy the inside content
    // newNode.innerHTML = node.innerHTML;

    // or

    // Move all child nodes at once
    newNode.append(...node.childNodes); // Spread all childNodes into append()

    // Replace node
    node.parentNode.replaceChild(newNode, node);

    return newNode;
}

let btn = document.querySelector('#btn');
replaceTag(btn, 'div');
```

Manipulating The CSS

Changing the Style (Manipulating the CSS):

1. Using attribute method

```
node.setAttribute('style', 'background-color: red');
node.setAttribute('class', 'darkMode')
```

2. Using style

```
node.style.backgroundColor = "red";
node.style.fontSize = "80px";
```

3. Using cssText property

```
node.style.cssText = 'background-color: black; color: white; font-size: 80px'
```

4. Using className property

```
node.className = 'lightMode'
```

5. Using clasList property : add(), remove(), toggle(), contains()

```
node.classList.add('class2');
node.classList.remove('class1');
node.classList.toggle('class2');
node.classList.contains('class1');
```

1. Create a simple website with theme changing functionality.

2. Solve first question by changing css class.

3. Solve the first question by toggle css class.

Event Handling

Event Handling:

An event in JavaScript is something that happens in the browser, like a user clicking a button or moving the mouse.

Example:

Mouse events (click, hover, double click, etc.)

Keyboard events (keypress, keyup, keydown, etc.)

Form events (submit, focus, input, etc.)

Note:

- You can monitor events using “monitorEvents(document)” and unmonitor using “unmonitorEvents(document)”.

We can handle event using following three methods:

1. Inline method (inline javascript)
2. Property method
3. Listener method

property-based event handling" or **"direct event assignment"**

```
node.event = () =>{
    // task
}
```

Note:

- ❑ you can't add multiple handlers for the same event on the same element using property method.

"event listener method" or "modern event handling"

```
div.addEventListener(event, () => {
    // task
})
```

How to remove event listeners:

If you want to properly add and then remove an event listener, you must:

1. Use addEventListener().

2. Use a named function, not an anonymous arrow function.

```
let random = () => {
    console.log("alpha")
}

div.addEventListener('click', random)

div.removeEventListener('click', random)
```

Event Object:

The event object is an object that is automatically passed to the event handler function when an event occurs.

It contains important information about the event, such as what triggered the event, the type of event, and other details like the mouse coordinates, key pressed, and more.

```
node.onclick = (eventObj) => {
    console.log(eventObj)
}
```

```
node.addEventListener('click',(eventObj)=>{
    console.log(eventObj)
})
```

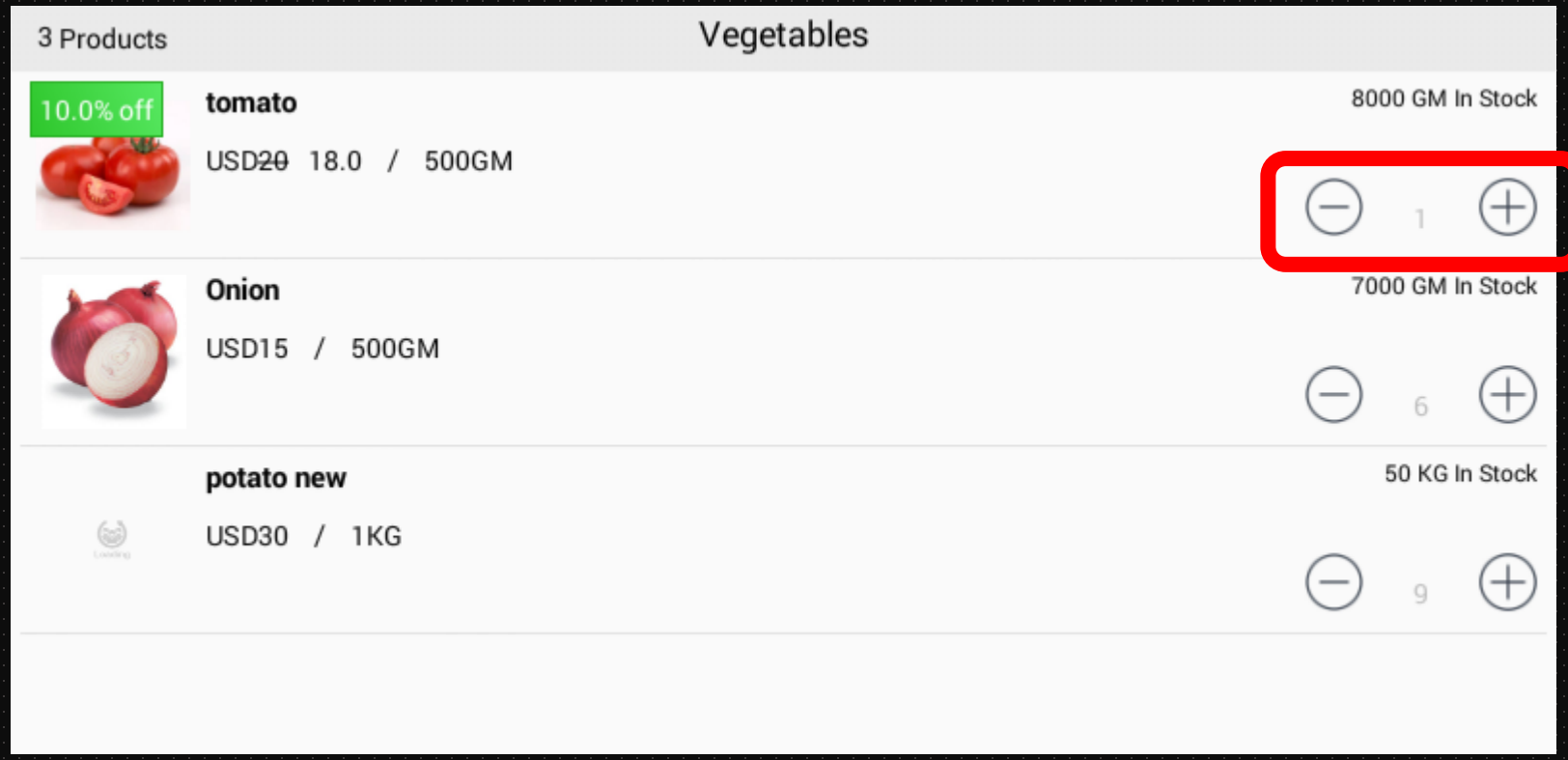
Most Used Events:

Category	Event	Role / Description
Mouse Events	click	Fired when an element is clicked.
	dblclick	Triggered by a double click.
	mousedown	When mouse button is pressed.
	mouseup	When mouse button is released.
	mouseenter	Mouse enters an element (no bubbling).
	mouseleave	Mouse leaves an element (no bubbling).
	mouseover	Mouse moves over an element or its children.
	mouseout	Mouse leaves an element or its children.
	mousemove	Mouse is moved within an element.
	contextmenu	Right mouse button is clicked.
Keyboard Events	keydown	Key is pressed down.
	keypress	(Deprecated) Key that produces a character is pressed.
	keyup	Key is released.

Category	Event	Role / Description
Form Events	submit	Form is submitted.
	reset	Form is reset.
	focus	Element receives focus.
	blur	Element loses focus.
	input	Value changes (real-time).
	change	Value of form element changes (on blur).
Touch Events	touchstart	Finger touches the screen.
	touchmove	Finger moves on screen.
	touchend	Finger is removed from screen.
	touchcancel	Touch interrupted (e.g., alert or system event).

Category	Event	Role / Description
Window Events	load	Page and resources fully loaded.
	DOMContentLoaded	DOM is fully loaded (without waiting for styles/images).
	resize	Window is resized.
	scroll	Page is scrolled.
Clipboard Events	copy	Content is copied.
	cut	Content is cut.
	paste	Content is pasted.

1. Build an increment–decrement counter similar to what you see in the shopping cart on Amazon or Flipkart.



2. Create a simple form and display the submitted details on the webpage. Ensure that if any field is left empty, the form should not be submitted.

Projects

(Next Lecture)