Week 2: JS + DOM + Web Audio

Speaker notes

Quick roadmap: review → DOM as objects → sliders/events → Web Audio basics → mixin

Review from Week 1

- `let`, `const`
- `console.log`
- `AudioContext`
- `OscillatorNode`
- `GainNode`

Speaker notes

Quick refresh so terms are top-of-mind.

Everything is an object

- Objects have **properties**
- Objects have **methods**

```
const obj = { x: 5, y: 10 };
console.log(obj.x);       // property
console.log(Object.keys(obj)); // method
```

Speaker notes

Anchor the mental model: we "get/set properties" and "call methods."

Variables

- Variables store objects (numbers, strings, AudioNodes…)
- `const` = cannot reassign
- `let` = can reassign

Variables allow us to keep track of and work with object.

```
const course = "CPM";
```

Speaker notes

Keep it simpleâ€ name, store, reuse.

Arithmetic Operators

Op.	Meaning	Code	Result
`+`	Addition	`2 + 3`	$2 + 3 = 5$
`-`	Subtraction	`7 - 4`	$7 - 4 = 3$
`*`	Multiplication	`6 * 2`	$6 \times 2 = 12$
`/`	Division	`10 / 4`	$\frac{10}{4} = 2.5$
`%`	Remainder	`10 % 4`	$10 \mod 4 = 2$

Speaker notes

Tie operators to pitch math (ratios & octaves).

Arithmetic

- Operators: `+ - * / % **`
- Useful for numbers, pitches, timing

```
const base = 220;
const fifth = base * (3/2); // 330
const octave = base * 2;
 // 440
console.log(fifth, octave);
```

What is the DOM?

- DOM = **Document Object Model**
- Browser turns HTML into a **tree of objects**
- `document` is our entry point into that tree
- Just like Web Audio nodes, it is an **object**
  - Properties (e.g. `document.body`)
  - Methods (e.g. `document.createElement()`)

Speaker notes

Stress that document is just another object we can use.

The DOM

```
// global Document object, root of the DOM
document;

// some things we can do with it:
document.getElementById("freq");              // method
document.body.style.backgroundColor = "pink"; // property
document.body.innerText = "Hello, Audio";     // property
```

HTML Tag Anatomy

```
<p>Hello</p>
```

- **Opening tag:** `<p>`
- **Closing tag:** `</p>`
- **Content:** `Hello`

Each tag becomes an **object** in the DOM with properties (like `.innerText`) and methods.

Hello

Speaker notes

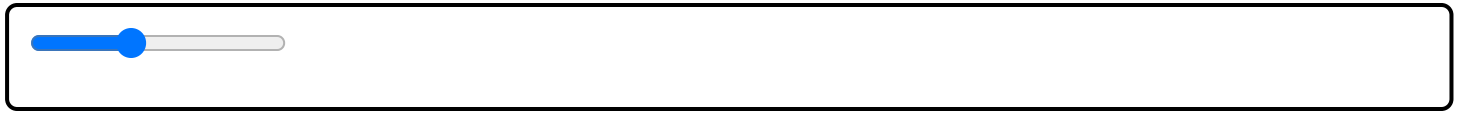
Link syntax to the idea that tags become JS-accessible objects.

HTML Tag Anatomy

```
<input id="freq" type="range" min="100" max="1000" value="440">
```

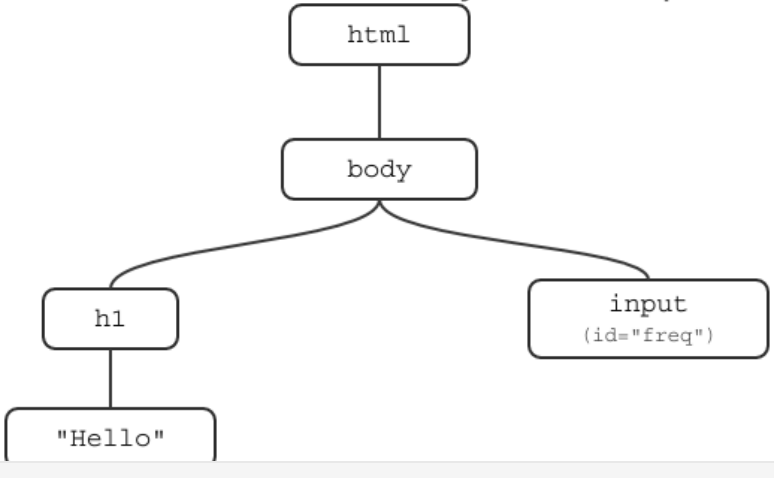
- **Attributes**: extra settings inside a tag, e.g. `id="freq"`, `type="range"`, `min="100"`, etc.
- **Self-closing tags**: don't wrap content, e.g. `<input ...>`

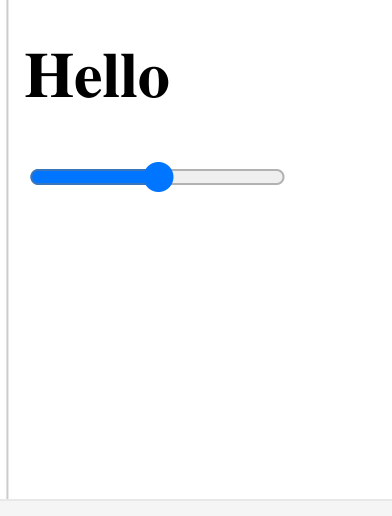
Each tag becomes an **object** in the DOM with properties (like `.value`) and methods.



DOM Tree Example

```
<html>
  <body>
      <h1>Hello</h1>
      <input id="freq" type="range" />
  </body>
</html>
```





Speaker notes

Show structure → objects; pair static diagram with a tiny live DOM.

DOM: Accessing Elements

- JavaScript needs a way to **find** elements in the DOM tree
- The `document` object gives us methods to do that
- `document.querySelector("CSS selector")`
  - Takes a **CSS-style selector string** (like `#id`, `.class`, `tagname`)
  - Returns the **first matching element object**
  - We will primarily use the `#id` method

Speaker notes

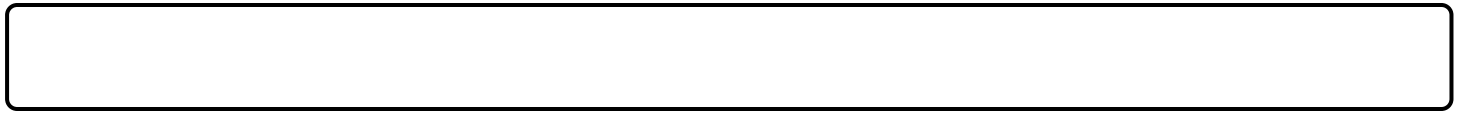
Emphasize "grab → store → then use properties and methods."

DOM: Accessing Elements

- To work with an element, first **grab it**
- Store it in a variable (just like `osc` or `gain`)

```
<input id="freq" type="range" min="100" max="1000" value="440">

<script>
  const slider = document.querySelector("#freq"); // grab object by id
  console.log(slider); // prints the <input> element
</script>
```

Open the console and take a look



DOM: Getting values

- `.value` gives the current slider position

```
<input id="freq" type="range" min="100" max="1000" value="440">

<script>
  const slider = document.querySelector("#freq");
  console.log(slider.value);          // "440" (string)
</script>
```

Open the console and take a look

Speaker notes

Input .value is a string; convert when you need math.

DOM: Converting values

```
console.log(typeof slider.value);          // "string"
console.log(typeof Number(slider.value));  // "number"
```

Open the console and take a look

Speaker notes

Point out type differences explicitly to avoid silent bugs.

DOM: Events

- Objects can "listen" for events
  - like input changes, button clicks, mouse hover, ...
- Add with `.addEventListener("input", ...)`

```
<input id="gain" type="range" min="0" max="1" step="0.01" value="0.2">

<script>
  const gainSlider = document.querySelector("#gain");

  gainSlider.addEventListener("input", () => {
    console.log("gain is now", Number(gainSlider.value));
  });
</script>
```

Open the console and take a look

Speaker notes

Basic UI loopâ€ read input, update text, repeat on events.

DOM: Events + Live Feedback

```
<input id="gain" type="range" min="0" max="1" step="0.01" value="0.2">
<span id="gainDisplay">0.2</span>

<script>
const gainSlider = document.querySelector("#gain");
const gainDisplay = document.querySelector("#gainDisplay");

gainDisplay.textContent = gainSlider.value;

gainSlider.addEventListener("input", () => {
  gainDisplay.textContent = gainSlider.value;
});
</script>
```

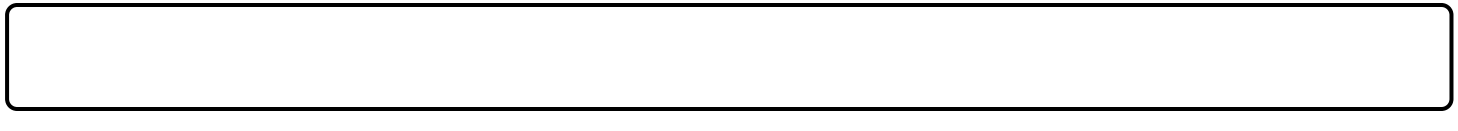
(We'll explain `() => {}` syntax next week)

Speaker notes

Basic UI loopâ€ read input, update text, repeat on events.

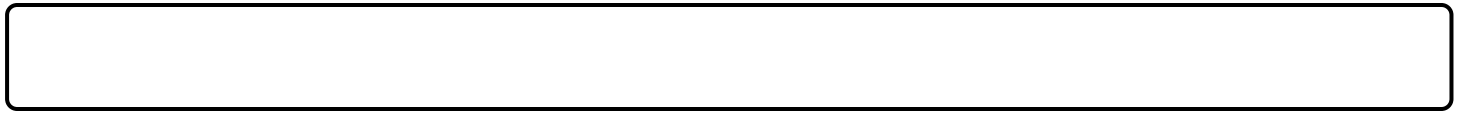
Web Audio: Simple Tone

```
<button id="start">Play</button>
<script>
    const ctx = new AudioContext();
    const osc = new OscillatorNode(ctx);
    const gain = new GainNode(ctx);
    gain.gain.value = 0.2;
    osc.connect(gain);
    gain.connect(ctx.destination);

    const startButton = document.querySelector("#start");
    startButton.addEventListener("click", () => {
      ctx.resume();
      osc.start();
    });
</script>
```



Properties vs Methods

```
osc.type = "triangle";              // property
osc.frequency.value = 165;      // property
gain.gain.value = 0.2;          // property

osc.connect(gain);              // method
gain.connect(ctx.destination); // method
osc.start();                    // method
```

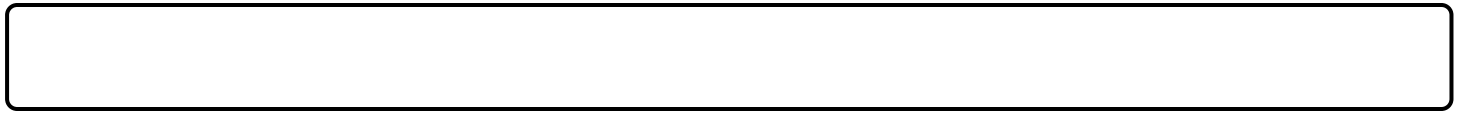


Speaker notes

Contrast properties (data you set) vs methods (actions you call).

Slider controls frequency

```
<input id="freq" type="range" min="100" max="1000" value="440">

<script>
 const ctx = new AudioContext();          // create AudioContext
 ctx.suspend();                           // prevent auto start

 const osc = new OscillatorNode(ctx);     // oscillator
 const gain = new GainNode(ctx);          // gain node
 gain.gain.value = 0.2;                   // set gain

 osc.connect(gain);                       // connect osc -> gain
 gain.connect(ctx.destination);           // connect gain -> speakers
 osc.start();                             // start oscillator

 const slider = document.querySelector("#freq"); // get slider element
```



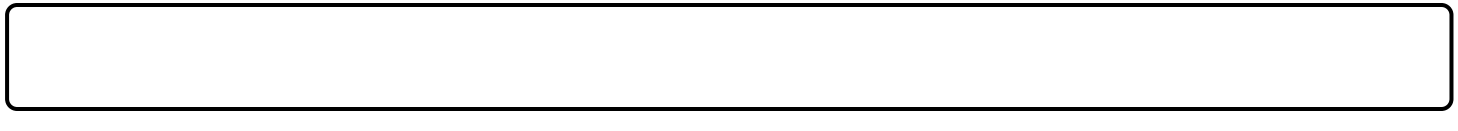
Slider controls gain

```
<input id="gain" type="range" min="0" max="1" step="0.01" value="0.2">

<script>
    const ctx = new AudioContext();          // create AudioContext
    ctx.suspend();                           // prevent auto start

    const osc = new OscillatorNode(ctx);     // oscillator
    const gain = new GainNode(ctx);          // gain node
    gain.gain.value = 0.2;                   // set initial gain

    osc.connect(gain);                       // connect osc &rarr; gain
    gain.connect(ctx.destination);           // connect gain &rarr; speakers
    osc.start();                             // start oscillator

    const slider = document.querySelector("#gain"); // get slider element
```

Mixing Nodes

- Many sources can connect to the same destination

```
const ctx = new AudioContext();
const master = new GainNode(ctx, );
master.gain.value = 0.125 // ~ -18 dBFS
master.connect(ctx.destination);

const osc1 = new OscillatorNode(ctx, { frequency: 110 });
const osc2 = new OscillatorNode(ctx, { frequency: 330 });
const osc3 = new OscillatorNode(ctx, { frequency: 513.33333 });

osc1.connect(master);
osc2.connect(master);
osc3.connect(master);

document.querySelector("button").addEventListener("click",  () => {
    ctx.resume();
```



Adding a Master Gain Slider

```
<button>Start</button>
<input id="masterGain" type="range" min="0" max="1" step="0.01" value="0">
<label id="gainDisplay" for="masterGain">0.0</label>
<script>
    const ctx = new AudioContext();
    const master = new GainNode(ctx, );
    master.gain.value = 0.0
    master.connect(ctx.destination);

    const osc1 = new OscillatorNode(ctx, { frequency: 110 });
    const osc2 = new OscillatorNode(ctx, { frequency: 330 });
    const osc3 = new OscillatorNode(ctx, { frequency: 513.33333 });

    osc1.connect(master);
    osc2.connect(master);
```

Wrap Up

- **JavaScript**: variables (`let`, `const`), arithmetic, `console.log`
- **DOM**: tree of objects, tags → objects, access with `document.querySelector`
- **Events**: sliders give values, `.value` is a property, update in real time
- **Web Audio**: `OscillatorNode`, `GainNode`, `.frequency.value`, `.gain.value`
- **Mixing**: multiple oscillators into one master gain
- **Big idea**: Everything is an object → properties & methods