Chapter 4: JavaScript in the Browser : DOM Manipulation:

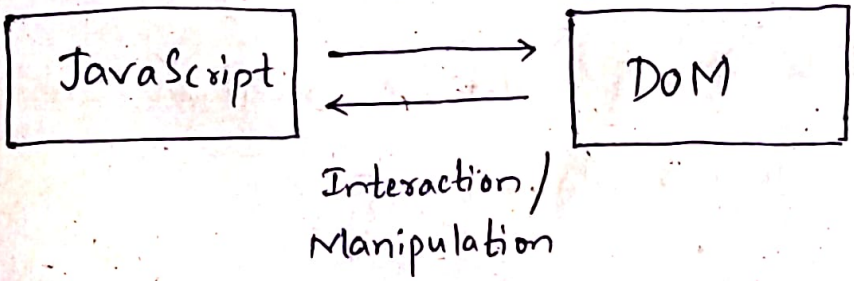
3. The DOM & DOM Manipulation:

→ DOM : Document Object Model

→ Structured representation of an HTML document.

→ The DOM is used to connect webpages to scripts like JavaScript.

→ For each HTML box, there is an object in the DOM that we can access and interact with.



Just remember that JavaScript and DOM are two different things.

We're going to use special JavaScript methods that allow us to interact and manipulate the DOM and therefore the

webpage and since we told methods, that means that they are functions attached to some objects. And that object is the document object. So this is the object that gives us access to the DOM.

The HTML webpage content is stored in the DOM which can then be accessed and manipulated by Javascript.

5. Project Setup & Details:

Import the starter project "4-DOM-Pig-Game" from Github. This project has all the HTML & CSS already written. We just need to write JavaScript.

6. First DOM Access & Manipulation:

app.js:

```
var scores, roundScore, activePlayer, dice;

scores = [0, 0];
```

```
roundScore = 0;

activePlayer = 1;

dice = Math.floor(Math.random() * 6) + 1;

document.querySelector('#current-' + activePlayer).
    .textContent = dice;

// document.querySelector('#current-' + activePlayer).
    textContent.innerHTML = '<em>' + dice + '</em>';

var x = document.querySelector("#score-0").
        textContent;
Console.log(x);

document.querySelector('.dice').style.display='none';
```

this converts decimal into integer

This generates a random number bn 0 and 1.

This is dynamic selection. It is based on activePlayer variable

This is just another way of doing the above thing.

This is getting

↙ This sets the display CSS property to none.

In this lecture, we have learnt how to:

- create our fundamental game variables.
- generate a random number.
- manipulate the DOM
- read from the DOM
- change CSS styles.

V.IMP

7. Events and Event Handling: Rolling the Dice:

What are Events?

→ Events: Notifications that are sent to notify the code that something happened on the webpage.

→ Examples: clicking a button; resizing a window, scrolling down or pressing a key.

→ Event listener: A function that performs an action based on a certain event. It waits for a specific event to happen.

So event listener is just a function that basically sits there and waits for a specific event to happen.

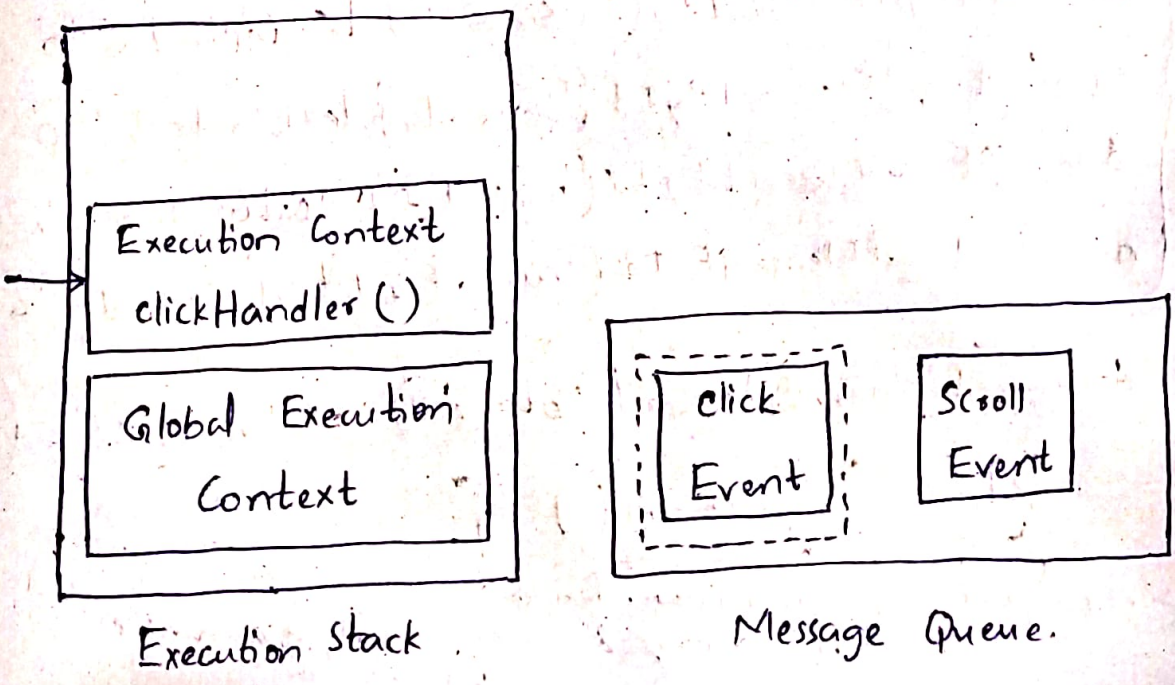
How are events processed?

First, we need to remember about the execution stack. The rule is that an event can only be processed or handled as soon as the execution stack is empty.

Which means, all the functions have returned.

Besides the execution stack, we also have something called the message queue in the Javascript engine. This is where all the events that happen in the browser are put. And they sit there, waiting to be processed. Which only happens once the Execution Stack is empty.

We can have an Event Listener which is a function that reacts to an event so that event listener is now called. And since it's a function, it gets its own execution context, which is then put on the top of the stack and becomes the active execution context



In this lecture, we will learn:

- How to set up an event handler
- What a callback function is
- What an anonymous function is
- Another way of selecting elements by ID
- How to change image in an <img> element.

app.js:

```
var scores, roundScore, activePlayer;

scores = [0, 0];
roundScore = 0;
activePlayer = 1;

document.querySelector('.dice').style.display = 'none';
```

just setting all scores to '0':

```
document.getElementById('score-0').textContent = '0';
document.getElementById('score-1').textContent = '0';
document.getElementById('current-0').textContent = '0';
document.getElementById('current-1').textContent = '0';

document.querySelector('.btn-roll').addEventListener(
    'click', function() {
```

'click': Listens for click event.

function(): anonymous function: It has no name & we can't call it anywhere but here.

```
// 1. Random number
var dice = Math.floor(Math.random()*6)+1;

// 2. Display the result
var diceDOM = document.querySelector('.dice');
diceDOM.style.display = 'block';
diceDOM.src = 'dice-' + dice + '.png';
```

selects image dynamically.

```
// 3. Update the round score IF the rolled
number was not a 1

});
```

8. Updating scores and Changing the Active Player:

In this lecture, we will learn:

- What the ternary operator is
- How to add, remove and toggle HTML classes.

app.js:

```
var scores, round, ...
:
:
```

```
// 3. Update the round score IF the rolled number was NOT a 1.
if (dice !== 1) {
    // Add score
    roundScore += dice;
    document.querySelector('#current-' + activePlayer).textContent = roundScore;
} else {
    // Next player
    activePlayer === 0 ? activePlayer = 1 : activePlayer = 0;
    roundScore = 0;

    document.getElementById('current-0').textContent = '0';
    document.getElementById('current-1').textContent = '0';

    document.querySelector('.player-0-panel').classList.toggle('active');
    document.querySelector('.player-1-panel').classList.toggle('active');
    document.querySelector('.dice').style.display = 'none';
}
});
```

Ternary operator. You know what this is.

9. Implementing Our 'Hold' Function and the DRY Principle:

What you will learn in this lecture:

- How to use functions to correctly apply the DRY (Don't Repeat Yourself) principle.
- How to think about the game logic like a programmer.

app.js:

```
var scores, ....
    :

// 3. Update the round score IF the rolled number was NOT a 1.
if (dice !== 1) {
    // Add score
    roundScore += dice;
    document.querySelector('#current-' + activePlayer).
        textContent = roundScore;
} else {
    // Next Player
    nextPlayer();  // This function is written below.
}
```

```
});

document.querySelector('.btn-hold').addEventListener(
    'click', function() {
    // Add current score to GLOBAL score
    scores[activePlayer] += roundScore;

    // Update the UI
    document.querySelector('#score-' + activePlayer).
        textContent = scores[activePlayer];

    // Check if player won the game
    if (scores[activePlayer] >= 20) {
        document.querySelector('#name-' + activePlayer).
            textContent = 'Winner!';
        document.querySelector('.dice').style.display='none';
        document.querySelector('.player-' + activePlayer +
            '-panel').classList.add('winner');
        document.querySelector('.player-' + activePlayer +
            '-panel').classList.remove('active');
    } else {
        //Next player
        nextPlayer();
    }
```

```
});

function nextPlayer() {
    //Next player
    activePlayer === 0 ? activePlayer = 1 : activePlayer
        = 0;
    roundScore = 0;

    document.getElementById('current-0').textContent
                                        = '0';
    document.getElementById('current-1').textContent
                                        = '0';

    document.querySelector('.player-0-panel').
        classList.toggle('active');
    document.querySelector('.player-1-panel').
        classList.toggle('active');
    document.querySelector('.dice').style.display = 'none';
}
```

10. Creating a Game Initialization Function:

app.js:

```
var scores, roundScore, activePlayer;
init(); → This function is written below.
```

```
document.querySelector('.btn-roll').addEventListener(
    'click', function() {

    //1. Random number
    .
    .
    .

function nextPlayer() {
    .
    .
    .
}

document.querySelector('.btn-new').addEventListener(
    'click', init);

function init() {
    scores = [0, 0];
    activePlayer = 0;
    roundScore = 0;
    document.querySelector('.dice').style.display = 'none';
    document.getElementById('score-0').textContent
                                        = '0';
    document.getElementById('score-1').textContent
                                        = '0';
```

```
document.getElementById('current-0').textContent = '0';
document.getElementById('current-1').textContent = '0';
document.getElementById('name-0').textContent = 'Player 1';
document.getElementById('name-1').textContent = 'Player 2';
document.querySelector('.player-0-panel').classList.
                remove('winner');
document.querySelector('.player-1-panel').classList.
                remove('winner');
document.querySelector('.player-0-panel').classList.
                remove('active');
document.querySelector('.player-1-panel').classList.
                remove('active');
document.querySelector('.player-0-panel').classList.
                add('active');
}
```

11. Finishing Touches: State Variables:

State Variable: A state variable simply tells us the condition of a system. We need a state variable when we need to remember something. In this state it is: is our game playing or not playing.

app.js:

```
var scores, roundScore, activePlayer, gamePlaying;

init();

document.querySelector('.btn-roll').addEventListener(
    'click', function() {
  if(gamePlaying) {
      //1. Random number
      var dice = .....
      :
      :

});

document.querySelector('.btn-hold').addEventListener(
    'click', function() {
  if(gamePlaying){
      //Add CURRENT score to GLOBAL score
      scores[activePlayer] += ....
      :
      if (scores[activePlayer] >= 20) {
          i
          gamePlaying = false;
      } else {
```

```
    //Next player
    nextPlayer();
  }
 }
});
```