

JavaScript

(part4)

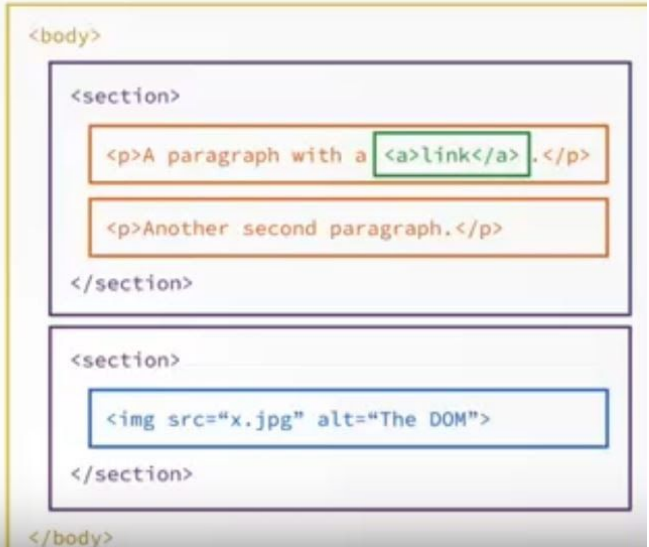
[Documentation for all web API's](#)

1. DOM manipulation: Making our JS interact with our webpage!

DOM stands for document object model

THE DOCUMENT OBJECT MODEL

- **DOM:** Document Object Model;
- Structured representation of an HTML document;
- The DOM is used to connect webpages to scripts like JavaScript;



The diagram illustrates the DOM tree structure. It shows a root node `<body>` which contains two child nodes, both `<section>` elements. The first `<section>` node contains two `<p>` (paragraph) nodes. The first `<p>` node contains the text "A paragraph with a" followed by an `<a>link` node, which is highlighted with a green box. The second `<p>` node contains the text "Another second paragraph.". The second `<section>` node contains an `` node, which is highlighted with a blue box. The entire structure is enclosed within the `<body>` root node.

2. Using the DOM concepts to make a pig game:

[Code for pig game](#)

Key points:

1. Use of Math function for calculating die number:
 $\text{dice} = \text{Math.floor}(\text{Math.random()} * 6) + 1$

2. Use of querySelector and textContent during DOM manipulation:

- querySelector: It helps us select elements just as we do in CSS using their class and ID.

Use as

```
document.querySelector('#ID_name').textContent= dice;
```

- textContent is used to change the value of selected ID.
- Selecting the ID numbers wisely:
The concept of type coercion was used while changing between players:
ID was selected as #ID_name-number. This number could be 0 or 1 and the variable activePlayer was used to keep track which player was active
- Problem with using textContent:
textContent just manipulates what's being shown on the screen but doesn't play around with the HTML. To resolve this innerHTML is used in place of textContent.

```
document.querySelector('#ID_name').innerHTML = '<em>' + dice + '</em>' ;
```


Thus the above line works as a setter because we set a value in this.

- DOM manipulation for changing CSS in JS:
`document.querySelector('ID/class').style.CSS_property = value;`

- Events

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.

Events are processed only when the execution stack is empty.

As soon as the execution stack is empty, the event queue is called and now this becomes the active execution stack.

[Event references](#)

`.addEventListener('type_of_event' , function to be called as event takes place);`

- Concept of the anonymous function:
`.addEventListener('type_of_event' , anonymous function);`
 This anonymous function can't be reused.
 Eg:
- `.addEventListener('click' , function(){`
 `//Do something here`
`});`
- Showing the correct image of dice:
 diceDOM was the variable used for the assigned query selector.
`diceDOM.src = 'dice-' + dice;`
- Using `getElementById` is used instead of the query selector because it loads faster.
`getElementById('score-0').textContent = '0';`
- Removing classes using query selector:
 To show an active dot in front of the inactive player:

```
document.querySelector('.player-0-panel').classList.remove('property_name');
```

Here property name was 'active'.

```
document.querySelector('.player-1-panel').classList.add('property_name');
```

However, this can be done in a better way using the toggle in which we can alternate without specifying:

```
document.querySelector('.player-0-panel').classList.toggle('property_name');  
document.querySelector('.player-1-panel').classList.toggle('property_name');
```

3. OOP's

Almost everything in JS is an object. Everything except primitive data types are objects.

The class in other programming languages are known as classes in constructor in JS.

THE OBJECT ORIENTED PARADIGM

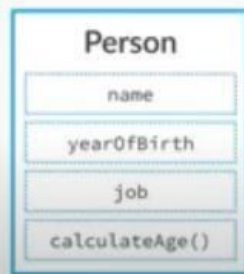
OBJECT-ORIENTED PROGRAMMING

- Objects interacting with one another through methods and properties;
- Used to store data, structure applications into modules and keeping code clean.

```
var john = {  
  name: 'John',  
  yearOfBirth: 1990,  
  isMarried: false  
};  
  
var jane = {  
  name: 'Jane',  
  yearOfBirth: 1969,  
  isMarried: true  
};  
  
var mark = {  
  name: 'Mark',  
  yearOfBirth: 1948,  
  isMarried: true  
};
```

Constructors acts as a blueprint where certain name value pairs can be assigned later on.

CONSTRUCTOR



INSTANCES

