Code Nation Master

JS and DOM

{CODENATION}™

DOM – Document Object Model

DOM

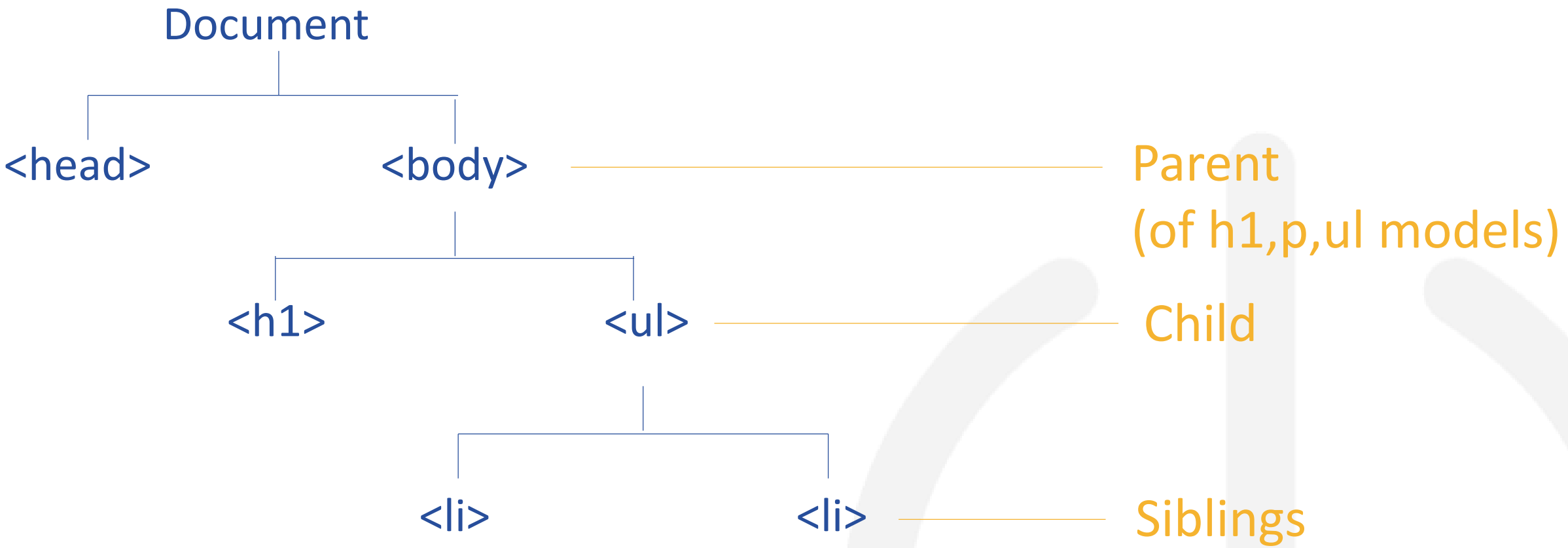
A representation of a webpage that JS can use,

Changes that JS makes to the DOM alter the webpage.

HTML structure

```
<html>
        <head>
                <title>Javascript and the DOM</title>
        </head>
        <body>
                <h1> Javascript and the DOM</h1>
                <p>Take control of the web page
                        <a href="#">link to a page</a>
                </p>
                <ul>
                        <li>First</li>
                        <li>Second</li>
                        <li>Third</li>
                </ul>
        </body>
</html>
```

DOM


Document
<head>
<body>
<h1>

Parent
(of h1,p,ul models)
Child
Siblings

Basic task JS can do with the DOM

- Select an element
- Read or change element
- Respond to user events

CODER
PUBLIC ALPHA

About Us FAQ Docs Blog

Log In

Sign Up

SIGN UP WITH GOOGLE

or

Username

Email

Phone Number

Password

We will text a verification code to your phone. You will be asked to enter the code on the next screen.

Message and data rates may apply. All user information is confidential. By signing up, you confirm that you accept the Terms of Use and Privacy Policy.

The Console

Activity

- Open first folder, index.html using Chrome
- Inspect, and click on ‘Console’

Activity: alert command

alert('Message me')

A browser function that shows a message window to the user.

Other commands

location.href

tells you the location of your file (locally)

window

shows you all the properties it contains, these are called ‘Window Object’

Clearing the console?

Ctrl + L

Activity: alert command

window.alert(“Message me”)

does the same thing as

alert(“Message me”)

Document object

Document object

Document is a global object representing the HTML and content of the page,

With JS you can select and control elements of the currently loaded web page

We can access DOM to:

- Select element on the page
- Manipulate elements
- Listening to user actions

getElementId

getElementById

document.getElementById('heading').style.color = "purple"

Change to different colours, then to black

getElementById

```
document.getElementById('heading').style.backgrou
ndColor = "yellow"
```

Change back to white

What's the point to use these commands?

We can:

- Click the headline to change headline colour
- Actions e.g. clicking, hovering over a button, scrolling or submitting a form

1AddEventListener

Activity: creating an app.js

- Inside your folder, create a new file called app.js

- In your html, add below your heading inside the body:

  <script src="app.js"></script>

Activity: Inside your app.js

```
const myHeading = document.getElementById('heading');

myHeading.addEventListener('click',()=>{
        myHeading.style.color='red';
        });
```

2Select Element by ID

Activity

In your html…

```
<h1 id="heading">THE DOM</h1>
<input id="input" type="text">
<button id="button">Submit</button>
```

Activity: storing the values

In your app.js, store the values of the heading in a const.

```
const heading = document.getElementById('heading');
```

Do the same for input and button

Store const for button

CHECK!

```
const input = document.getElementById('input');
const button = document.getElementById('button');
```

Activity

Make the button listen for mouse clicks using the correct method

Then, get the value from the input value and use that to change the colour of the heading

```
button.addEventListener('click',()=>{
    heading.style.color = input.value;
});
```

3Select Elements by particular type

document.getElementsByTagName

Example: Get elements by tag name

```
//=><ul id="a">
const el = document.getElementId('a');

//=>[<p>,<p>,<p>]
const els = document.getElementsByTagName('p');
```

Example: Selecting an element by index

```
const els = document.getElementsByTagName('p');

Let el = els[0];
```

Example: Select each element using loop

```
const els = document.getElementsByTagName('p');

for (let i = 0;  i < els.length; i ++){
        els[i]
}
```

Activity: looking at html

In your html, you have the heading “THE DOM”, and four items on the list:

- C
- O
- D
- E

Activity: Using the console

Try each of the below in your console:

```
const list = document.getElementsByTagName('li')
list.length
list[0]
list[3]
list[0].style.color='red'
```

Activity: app.js

Target the list elements and store in a constant, add to your app.js

```
const list = document.getElementsByTagName('li');
```

You may want to tell the console to log it so we can see if it's doing the job

```
console.log(list);
console.log(list.length);
```

Activity: change color of each element

Add this to your app.js

```
for (let i = 0; i< list.length; i++){
        list[i].style.color = ‘orange’;
}
```

Activity

Below your code, by using **document.getElementsByClassName('not-orange'),**

Repeat the same steps to change these to 'red'

Solution

```
const notOrange = document.getElementsByClassName('not-orange');

for (let i = 0; i< notOrange.length; i++){
        notOrange[i].style.color = 'red';
}
```

4 querySelector querySelectorAll

Activity: HTML Console

Open the index.html and try each of these in your HTML console:

document.querySelectorAll('li')
document.querySelector('li')

document.querySelector('#heading')

document.querySelector('.list-parent')
document.querySelectorAll('.green')

So these code are functionally identical

```
const myElement = document.getElementById('myId');
const myElement = document.querySelector('#myId');
```

Activity: app.js

Think about what would be output be before typing these to app.js and open html and its console:

```
const listItems = document.querySelectorAll('li:nth-child(even)');

console.log(listItems);
console.log(listItems.length);
```

Activity: Iterate over selected items

```
for(let i=0; i < listItems.length; i++){
        listItems[i].style.color = 'lightgreen';
}
```

Challenge

Given challenge.html and js:

Cycle over the list items and apply colours from the array called colours

Folder 4 challenge

5 textContent and innerHTML

Activity: Storing elements in const

Inside the app.js, set the const as below:

```
const placeholder = document.getElementById('placeholder');
const input = document.getElementById('input');
const submit = document.getElementById('submit');
const list = document.getElementById('list');
```

Activity: textContent

We want the text input to be placed in the heading, we can attach addEventListener to the button.

```
submit.addEventListener('click', () =>{
        placeholder.style.color ='goldenrod';
        placeholder.textContent = input.value;
}
```

Activity: use of innerHTML

Comment out previous and do this instead:

```
submit.addEventListener('click', () =>{
        placeholder.style.color ='goldenrod';
        placeholder.textContent = `<li>${input.value}</li>`;

        list.innerHTML = `<li>${input.value}</li>`;
}
```

Can you explain the difference?

6 Changing Element Attributes

Activity: store image element in a const

```
const image = document.getElementById('image');

console.log(image);
```

Activity: back to browser

Before continuing the app.js, we want to try in the **console** first.

Open index.html, open console and type:

image.src = ‘link to image here’,

In my case

Image.src=‘https://www.mrmen.com/wp-content/uploads/2017/03/springbanner-small-1.jpg’

Activity: set const and function

```
const input = document.getElementById('input');
const button = document.getElementById('submit');

button.addEventListener('click', ()=>{
        image.src = input.value;
})
```

7 Styling Elements

Activity: viewing the image style

First up we want to check out the image style by giving the img a const.

```
const image = document.getElementById('cat');
console.log(image.style);
```

Activity: show and hide image

We want to hide the image when the button is clicked, and show when it is clicked again, so we need to set another const and then a function, add code inside the function.

```
const button= document.getElementById('submit');

button.addEventListener('click', () => {
 //your code here
}
```

Solution

```
button.addEventListener('click', () => {
        if(image.style.display == 'none') {
                image.style.display = 'block';
                button.textContent = 'hide';
        } else {
                image.style.display = 'none';
                button.textContent = 'show';
        }
})
```

8 Create New Elements

Activity: set input and button as const

```
const input = document.getElementById('input');
const button = document.getElementById('submit');
```

Then create a list item when the person presses the submit button

```
button.addEventListener('click', ()=> {
        let listItem = document.createElement('li');
        listItem.textContent = input.value;
})
```

Activity: show the updated list

First up we need to create a new let for the list

```
let list = document.getElementByTagName('ul')[0];
```

Then add the following inside the function

```
list.appendchild(listItem);
```

Challenge

- Clear the input field when user presses submit
- Add feature where user can show/hide the list

Solution

- Clear the input field when user presses submit

```
input.value = ‘ ’;
```

Solution

- Add feature where user can show/hide the list
  - Add a new button in html

    ```
    <button id='showhide-btn'>hide</button>
    ```
  - In app.js, add a new const

    ```
    const showhidebtn=document.getElementbyId('showhide-btn');
    ```
  - Add a function

    ```
    showhidebtn.addEventListener('click', () => {
            let list = document.getElementByTagName('ul')[0];
            if(list.style.display == 'none') {
                    list.style.display = 'block';
                    showhidebtn.textContent = 'hide';
            } else {
                    image.style.display = 'none';
                    showhidebtn.textContent = 'show';                    }})
    ```

9 Removing elements

Activity: Remove last item

Think about the steps you need:

- Set a new const for the remove button
- Create a new function to remove last item when button is clicked, using the last child method:

  node.removeChild(childElement);

  Node in this case represent the list of items<li>, so last child would be: **li:last-child**

Solution

- Set a new const for the remove button

```
const removeBtn = document.getElementById('remove');
```

- Create a new function to remove last item when button is clicked, using the last child method:

```
removeBtn.addEventListener('click',() =>{
        let listItem = document.querySelector('li:last-child');
        let list = document.getElementsByTagName('ul')[0];
        list.removeChild(listItem);
}
```