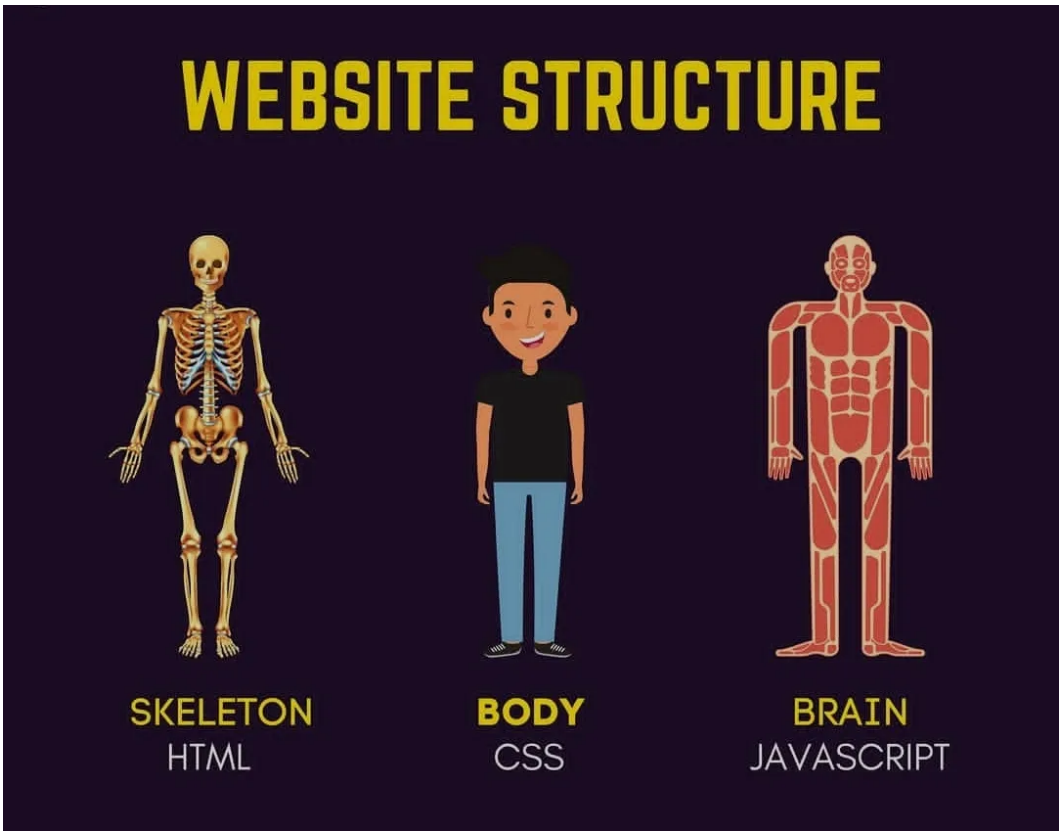
Class-3: Intro to DOM

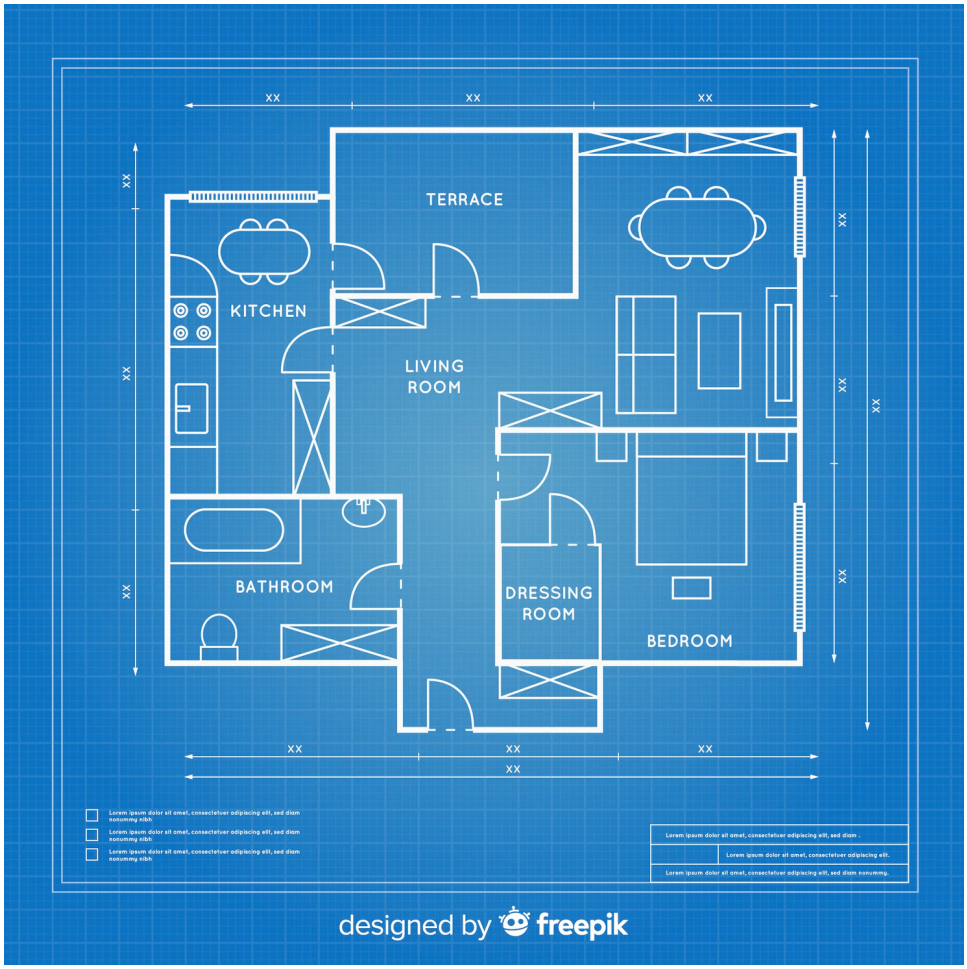
Introducing JS

Instructor Task (5 mins): HTML vs CSS vs JS

- Let’s take analogy of car with HTML
  - Car body : Structure of HTML
  - Car color : Styling (CSS)
  - Gears, breaks : Functionality (JS)



  - DOM is blueprint of house. Its not the actual house. But using it you can plan changes like make the bedroom 16 ft size, color the living room red etc, and through DOM API those changes can be executed.



- While searching on google, some of the DOM nodes will be formed dynamically, this is because DOM tree is getting updated based on your search (show it on live)

What is DOM?

- Imagine this🤔 you are watching TV and you don't like the show that's being streamed, and you want to change it and you also want to increase its volume.To do that, there has to be a way for you to interact with your television.So how do you control your tv now?
- By using `remote`

- The remote serves as the **bridge** which allows you interact with your television.
- In the same way, JavaScript helps you to interact with the HTML page via **DOM.**
- For example, say that you want a button to change colours when it gets clicked or an image to slide when the mouse hovers over it. First, you need to reference those elements from your JavaScript.

Jab HTML met JS 🤝

How to add JS inside HTML

```
<html>
<head>
  <title>Connecting JS</title>
</head>
<body>
</body>
</html>

<script type="text/javascript">
    function doSomething(){
         // Your Code Goes Here
    }
</script>
```

- Inside script tag you can write your js code

Console.log():

- console.log() in javascript is your best friend life long

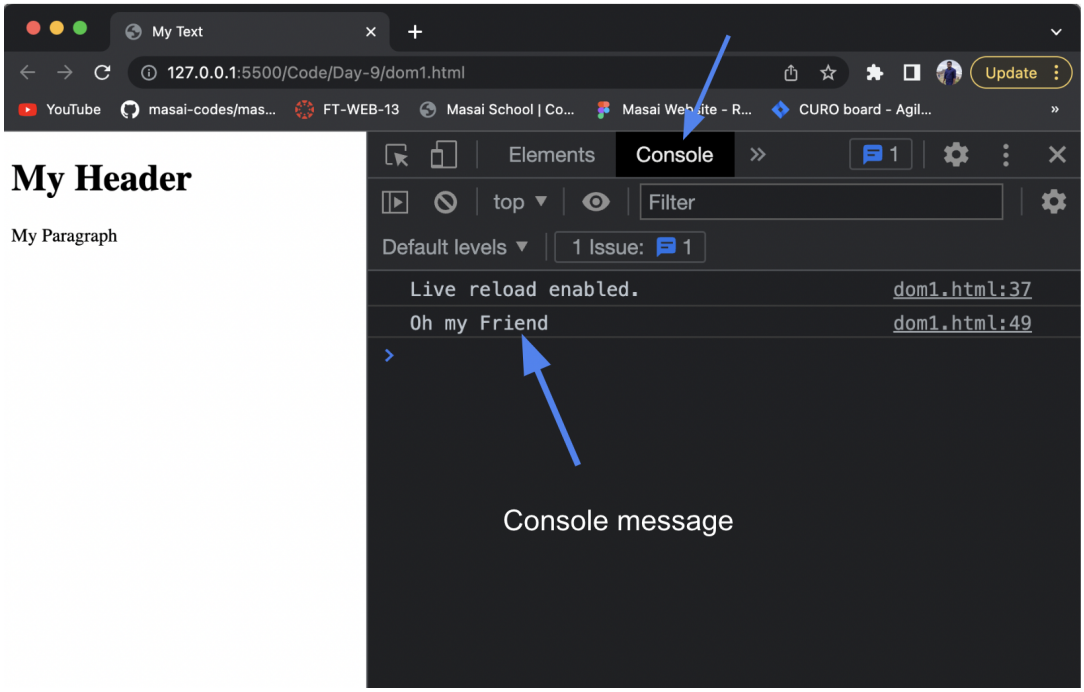
Where to find your best friend ?

- As you already know, you should use console inside your js code, so I am writing console.log() in script tag

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Text</title>
  </head>
  <body>
    <h1>My Header</h1>
    <p>My Paragraph</p>
  </body>
</html>
```

```
<script>
    console.log("Oh my Friend")
</script>
```

- Now, where can I find my best friend
  - Step-1: Run HTML file on browser.
  - Step-2: Right click and click on inspect.
  - Step-3: You can see console beside Elements

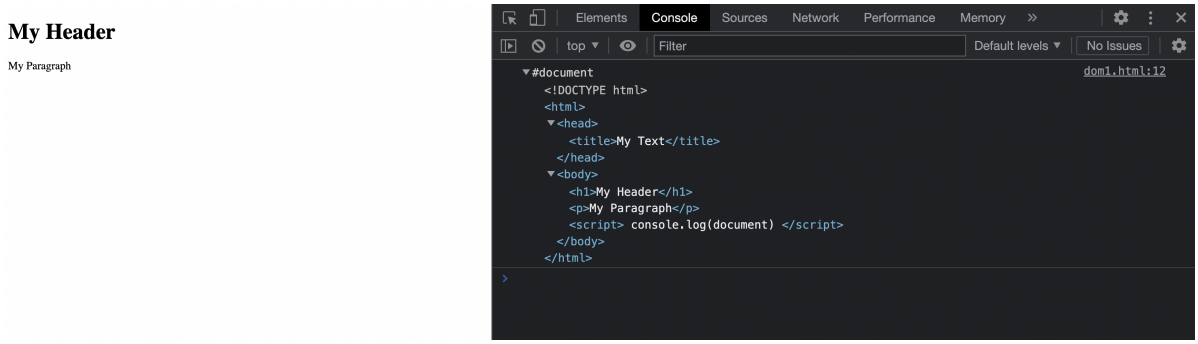


- Now with help of your friend let's see what is hiding in our document

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Text</title>
  </head>
  <body>
```

```
    <h1>My Header</h1>
    <p>My Paragraph</p>
  </body>
</html>
<script>
    console.log(document)
</script>
```

Output

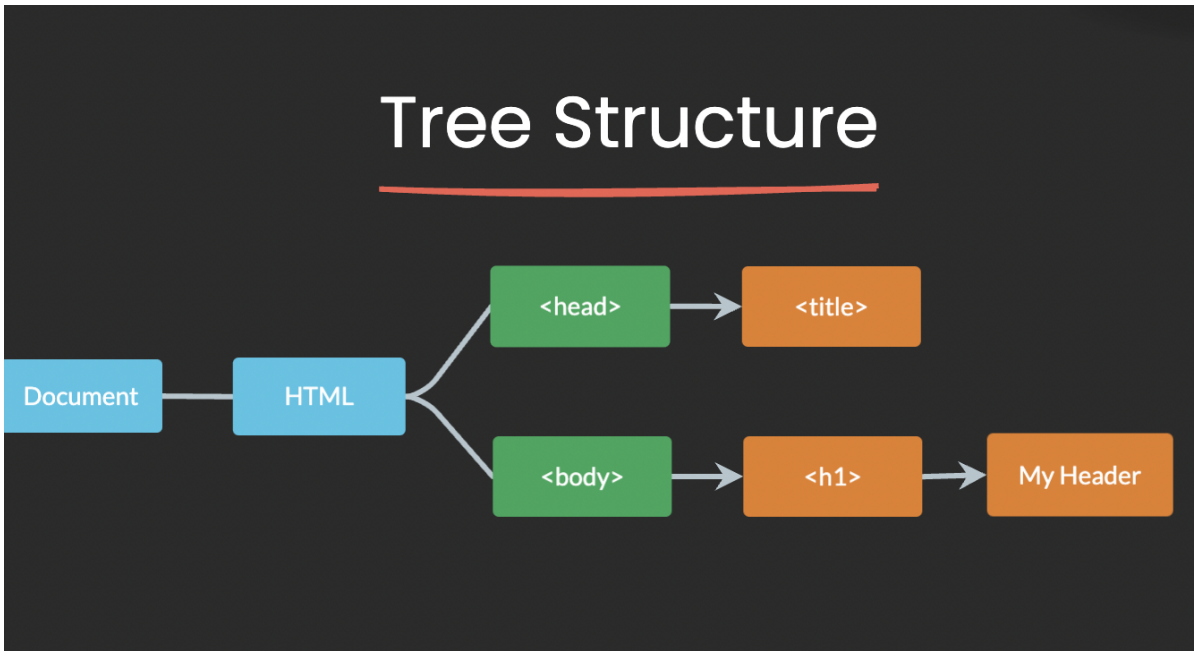


- The DOM is a tree-like representation of the web page that gets loaded into the browser.

- Whenever html document is loaded on browser, corresponding to that document another representation of document is created which is known as DOM.
- Let's suppose you have this code

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Text</title>
  </head>
  <body>
    <h1>My Header</h1>
  </body>
</html>
```

- If you open this in browser the web browser creates a DOM of the webpage when the page is loaded. The DOM model is created as a tree of objects like this



- Each entity in the tree is called a *node*. There are two types of nodes:

- Those (in green here) that correspond to HTML tags like `<body>` . These nodes are called element nodes and they can have subnodes, called *child nodes* or children. Those (in orange) that match the textual content of the page. These nodes are called *text nodes* and do not have children.
- Node can have many children but only one parent.
- If an object is removed from DOM, so are all its children (like cutting off a branch from the tree all subbranches will also fall)

Student Task:

- Ask them to install extension and see the tree structure of any website

Access the DOM with the document variable:

- So `console.log(document)` returning entire document in object format
- Write the following out in any page and see what the response is

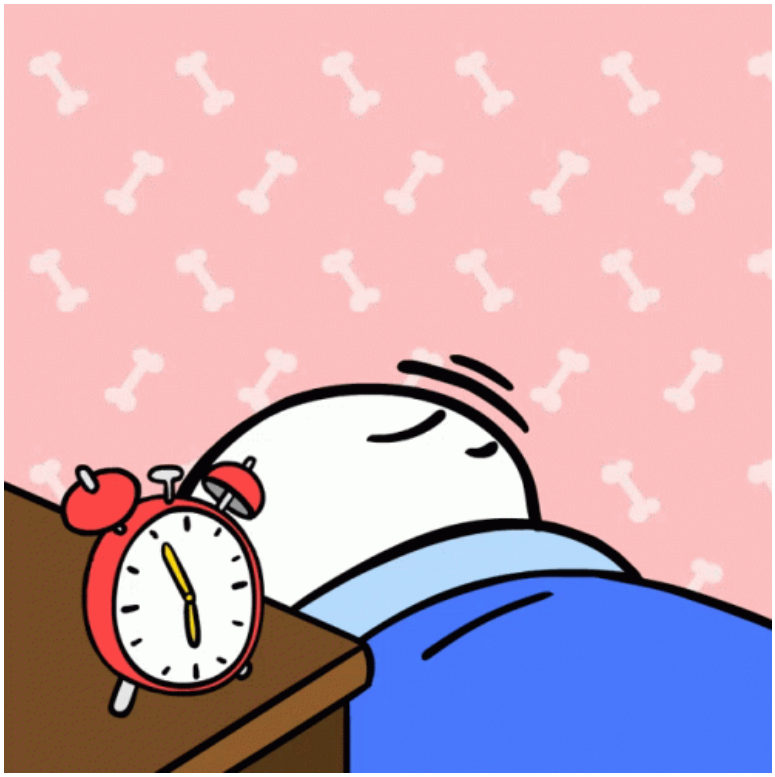
```
// URL of a page
console.log(document.URL)
// title of a page
console.log(document.title)
// domain
console.log(document.domain)
// doctype
console.log(document.doctype)
// head
console.log(document.head)
// body
console.log(document.body)
// all elements
console.log(document.all)
// forms
console.log(document.forms)
// links
console.log(document.links)
```

```
// images
console.log(document.images)

console.log(document.head); // "h" variable contains the c
ontents of the DOM's head

console.log(document.body); // "b" variable contains the c
ontents of the DOM's body
```

EVENTS

- JS interacts with HTML through Events. An event occurs when the user interacts with a page (or the browser changes a page).
- In real life we have all responded to Events - remember the morning school bell! We all responded to the “bell event” by rushing to our classes.

- When an event occurs on an object, we respond to it with an action.



- Here, Alarm Clock is an object
- Event to **ring** is set, lets say at 7am
- When it rings we respond by waking up (hopefully!!).

- Similarly in JS we have many events to which our App need to respond

- Similarly, in the above example button is an object
- Event of **click** is set
- When its clicked the page responds by showing “Button Clicked”

Let’s see how to code a HTML element to respond when clicked.

onClick Event:

- The `onclick` event occurs when the user clicks on an element.
- The `onclick` event executes a certain functionality when a button/element is clicked.

Basic `onclick` syntax:

```
<element onclick="functionToExecute()">Click</element>
```

Example:

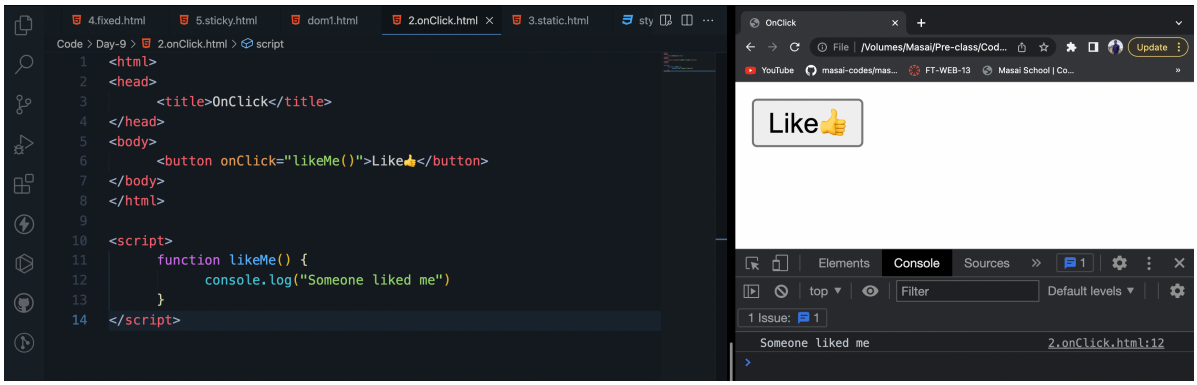
```
<html>
<head>
      <title>OnClick</title>
</head>
<body>
      <button onClick="likeMe()">Like👍</button>
</body>
</html>
```

- Now we need to define this function in javascript to make it work

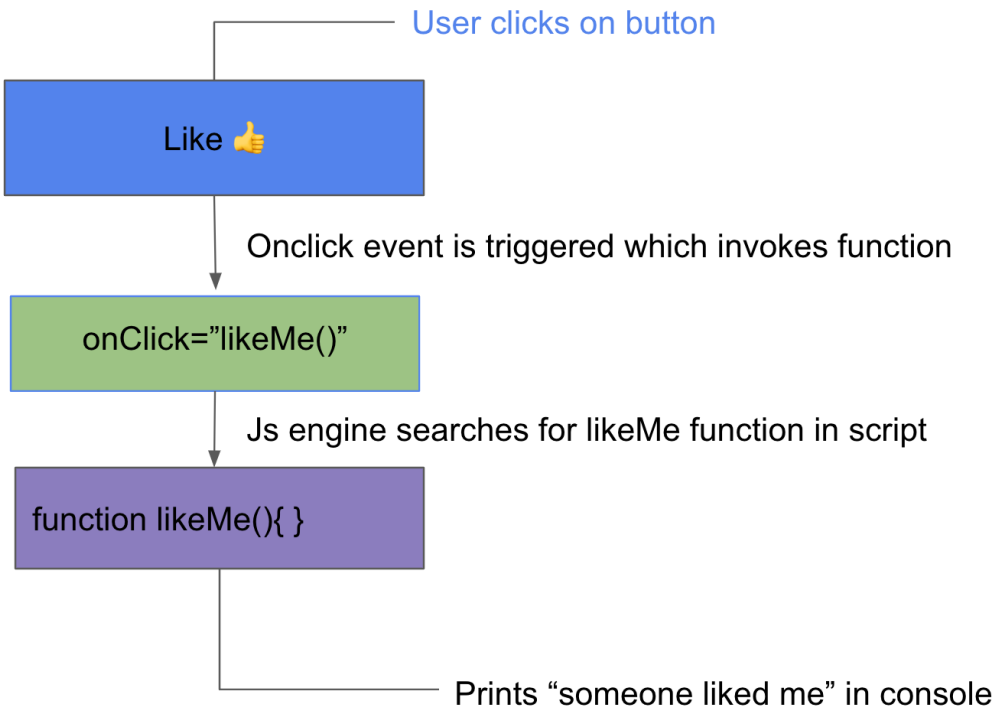
```
<script>
      function likeMe() {
            console.log("Someone liked me")
```

```
        }
</script>
```

- Now when we click on “Like” button, we will get `“Someone liked me”` in console



- Let’s dry run it

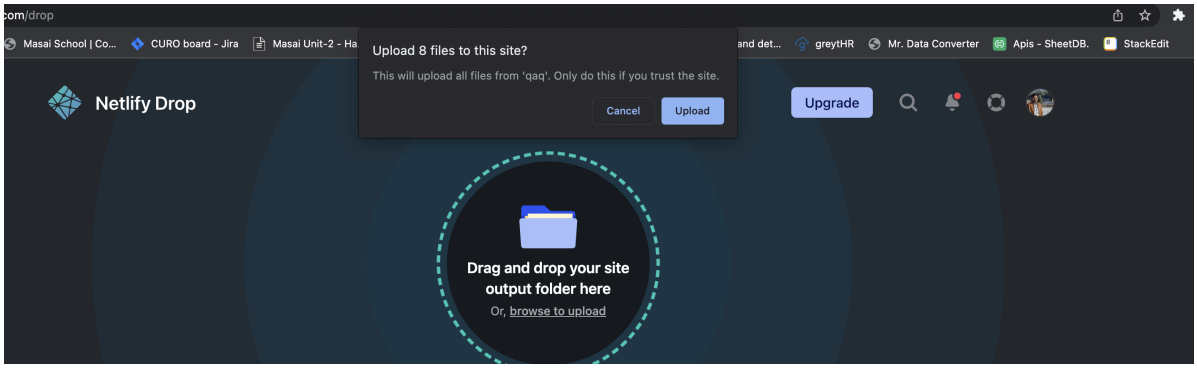


Student Task:

- https://course.masaischool.com/problems/26933

Alert():

- The **alert()** method in JavaScript is used to display a virtual alert box.
- It is mostly used to give a warning message to the users. It displays an alert dialog box that consists of some specified message (which is optional) and an OK button.



- When the dialog box pops up, we have to click "OK" to proceed.

Syntax

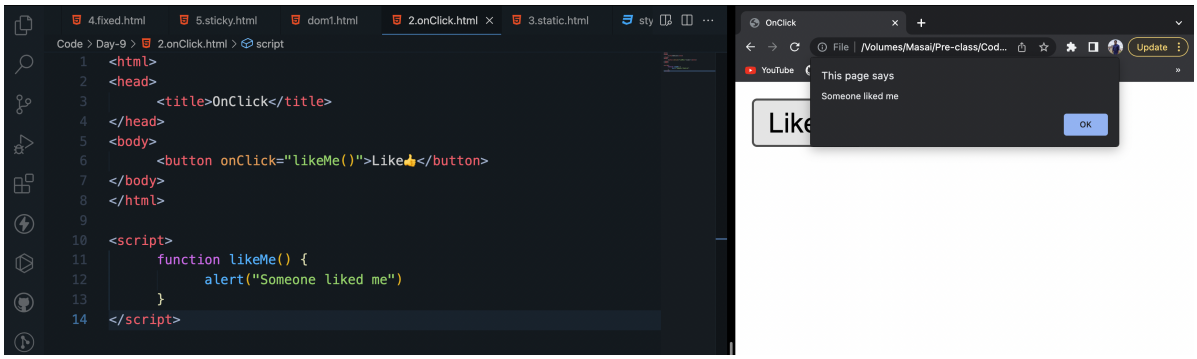
```
alert(message)
```

- In the above example instead of console, we will try to keep alert

```
<html>
<head>
      <title>OnClick</title>
</head>
<body>
      <button onClick="likeMe()">Like👍</button>
```

```
</body>
</html>

<script>
      function likeMe() {
            alert("Someone liked me")
      }
</script>
```

Output



Student task:

https://course.masaischool.com/problems/26934

document.getElementById()

Syntax

```
document.getElementById(id_Name)
```

Parameters

- `id` The ID of the element to locate. The ID is case-sensitive string which is unique within the document; only one element may have any given ID.

Return value

- An `Element` object describing the DOM element object matching the specified ID, or `null` if no matching element was found in the document.
- The `Document` method `getElementById()` returns an `Element` object representing the element whose `id` property matches the specified string.

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
    <p id="dhoni">I am Dhoni from Chennai</p>
    <p id="kohli">I am Kohli from Banglore</p>
  </body>
</html>

<script>
  let msd = document.getElementById("dhoni");
  console.log(msd);

  let chiku = document.getElementById("kohli");
  console.log(chiku)
</script>
```

Output:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
    <p id="dhoni">I am Dhoni from Chennai</p>
    <p id="kohli">I am Kohli from Banglore</p>
  </body>
</html>

<script>
  let msd = document.getElementById("dhoni");
  console.log(msd);

  let chiku = document.getElementById("kohli");
  console.log(chiku)
</script>
```

I am Dhoni from Chennai

I am Kohli from Banglore

Elements Console

top Filter

Default levels No Issues

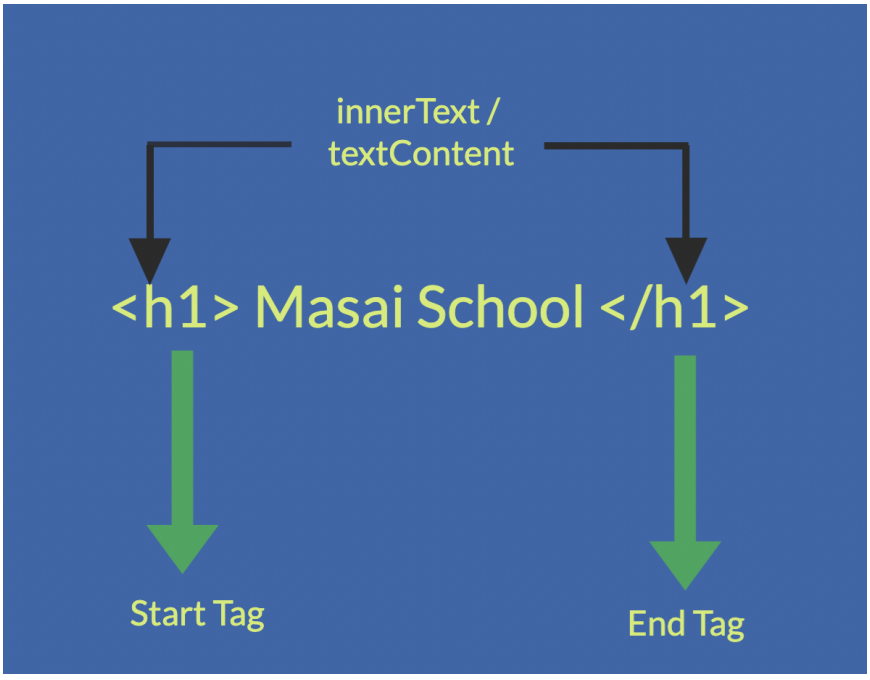
1.html:14
I am Dhoni from Chennai

1.html:17
I am Kohli from Banglore

Read more : https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById

innerText / textContent:

- Did you remember in our first class we have discussed about `HTML elements`



- The content inside opening and closing tag is called as `innerText` also called as `textContent`
- The `innerText` property sets or returns the text content of an element.

Syntax

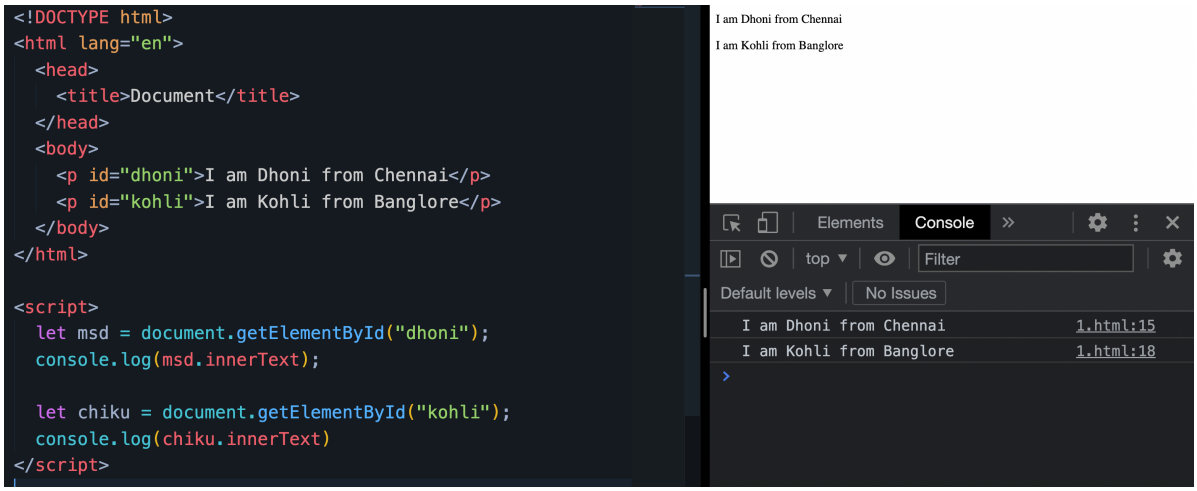
```
element.innerText
```

Example:

Let's take previous example and try to get innerText of tags

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
    <p id="dhoni">I am Dhoni from Chennai</p>
    <p id="kohli">I am Kohli from Banglore</p>
  </body>
</html>

<script>
  let msd = document.getElementById("dhoni");
  console.log(msd.innerText);

  let chiku = document.getElementById("kohli");
  console.log(chiku.innerText)
</script>
```

Output:



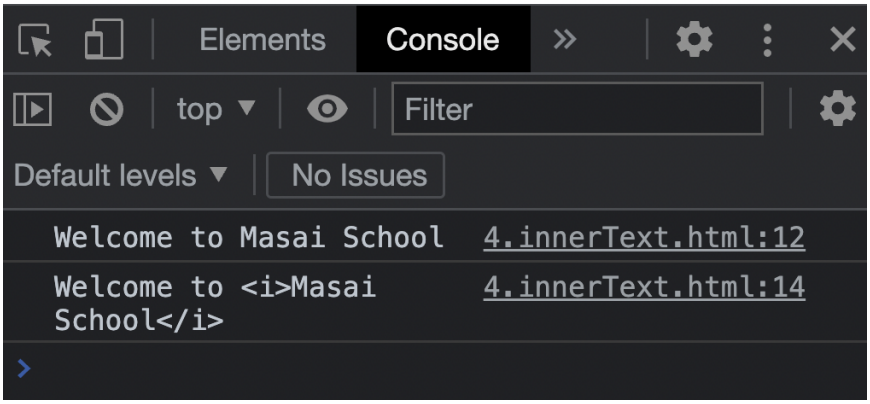
Example:

`<i>` tag is italic tag

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>innerText vs innerHTML</title>
  </head>
  <body>
    <div id="box">Welcome to <i>Masai School</i></div>
  </body>
</html>

<script>
      console.log(document.getElementById("box").innerTex
t)

      console.log(document.getElementById("box").innerHTM
L)

</script>
```

Output:

Welcome to *Masai School*



- From the above example you can clearly see that
  - `innerText` is retreiving only text inside tags
  - `innerHTML` is retreiving with HTML elements (<i> tag)

Student Task

https://course.masaischool.com/problems/26936

Changing innerText of an element

- Javascript provides us with the `innerText` property that we can use to change the text inside an element.

Syntax:

```
element.textContent = "new_value"
```

Example:

```
<html>
<body>
  <h1>Heading 1</h1>
</body>
<script>
  // change the text content in the heading
  document.querySelector("h1").textContent = "H1";
</script>
</html>
```

Student task:

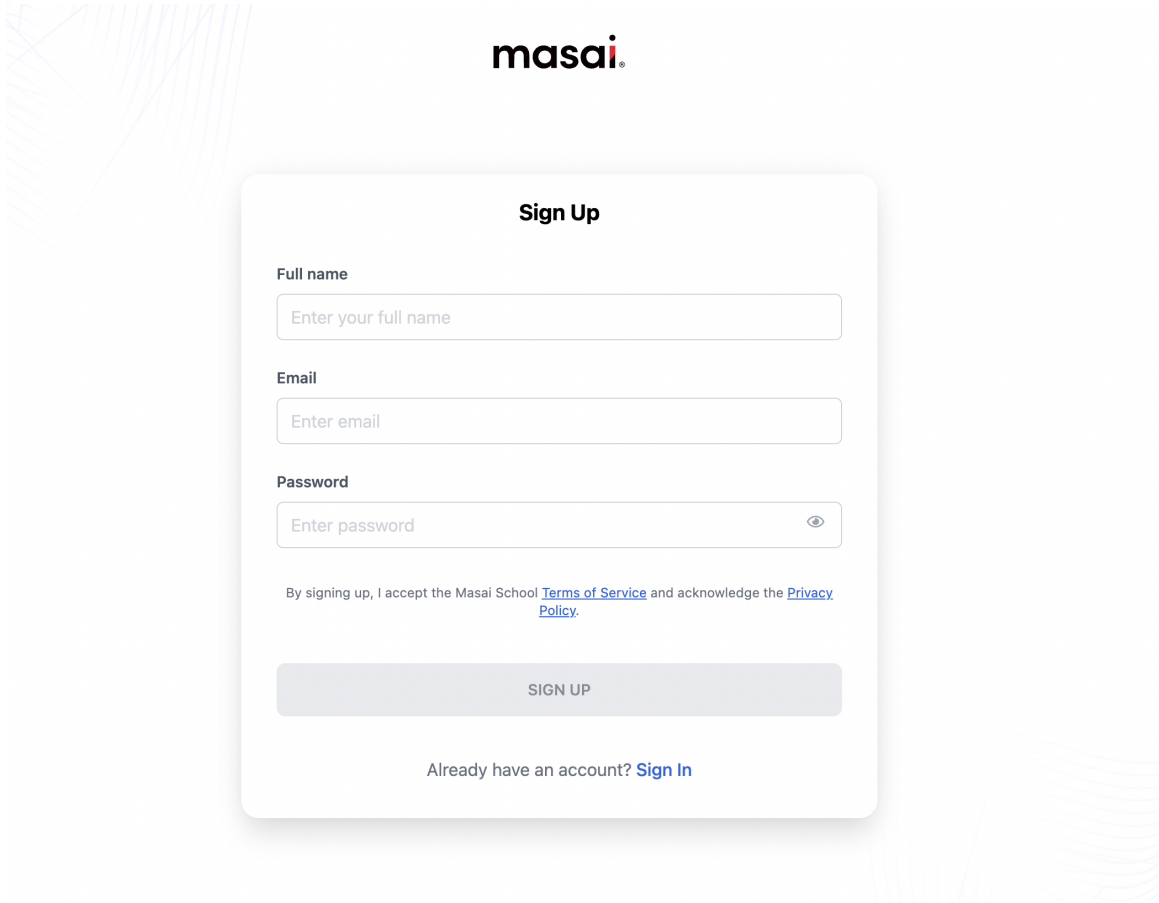
https://course.masaischool.com/problems/26938

Example 2 : Codepen

Value:

Getting values from Input tags

- In HTML introduction we have seen so many input tags.
- We have also seen masai form



- To interact with user, we need to get the value of
  - Full name
  - Email
  - Password
- How to get those values?
  - For getting those values we have an `attribute` in input tag known as `value` which is by default empty.

```
<input type="text" value=""/>
```

- Try to type in some value and check the output

```
<input type="text" value="Masai School"/>
```

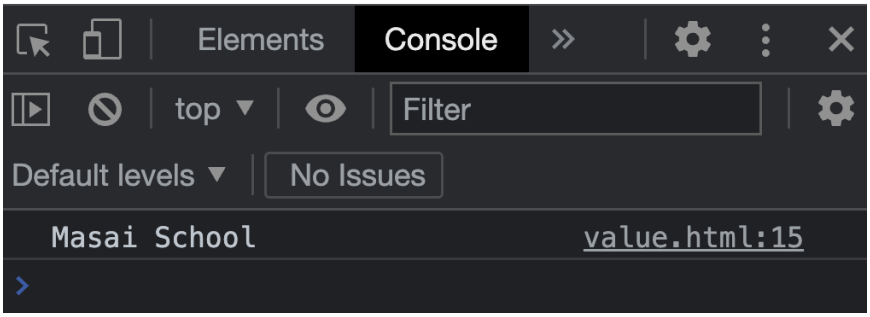
Output:

- So, whatever we type in that input box, we can capture that in `value` attribute.

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
    <input id="name" type="text" value="" />
    <button onClick="catchValue()">Submit</button>
  </body>
</html>

<script>
  function catchValue() {
    var studentName = document.getElementById("name").valu
e;
    console.log(studentName);
  }
</script>
```

Output:



Live code : Codepen

Example 2 : Codepen

Getting values from Select tags

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" /
>
    <meta name="viewport" content="width=device-width, ini
tial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <select id="gender">
```

```
      <option value="">Select gender</option>
      <option value="MALE">Male</option>
      <option value="FEMALE">Female</option>
    </select>
    <button onClick="getGender()">Submit</button>
  </body>
</html>

<script>
  function getGender() {
    var gen = document.getElementById("gender").value;
    console.log(gen);
  }
</script>
```

Live code : Codepen

getElementsByClassName():

- The `getElementsByClassName()` method returns a collection of elements with a specified class name(s).
- The `getElementsByClassName()` method returns an array-like collection (list) of HTML elements.

Syntax:

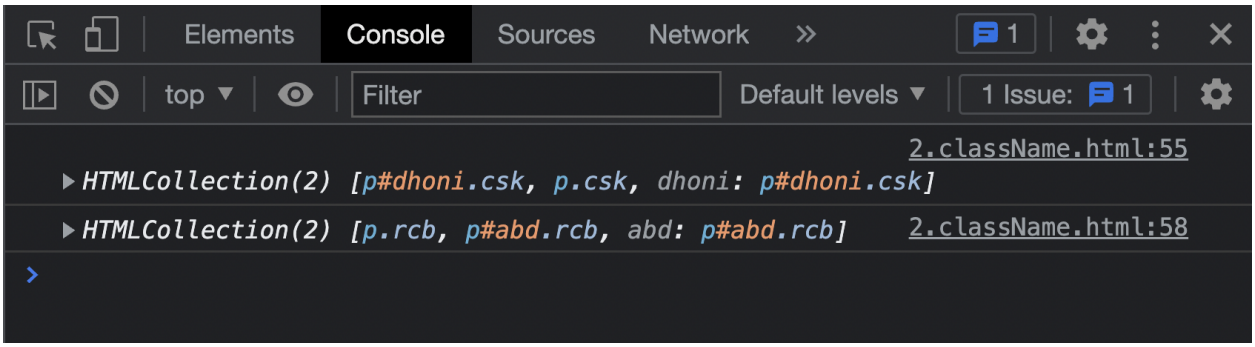
```
document.getElementsByClassName(classname)
```

Example:

```
<html lang="en">
  <head>
    <title>Document</title>
```

```
  </head>
  <body>
    <p class="csk" id="dhoni">I am Dhoni from Chennai</p>
    <p class="mi">I am Sachin from Mumbai</p>
    <p class="rcb">I am Kohli from Banglore</p>
    <p class="mi">I am Rohit from Mumbai</p>
    <p class="rcb" id="abd">I am ABD from Banglore</p>
    <p class="csk">I am Raina from Chennai</p>
  </body>
</html>

<script>
  var chennai = document.getElementsByClassName("csk");
  console.log(chennai);

  var banglore = document.getElementsByClassName("rcb");
  console.log(banglore);
</script>
```

Output:



Live code: Codepen

- The length Property returns the number of elements in the collection.

```
  console.log(chennai.length) //2
```

- The elements in a collection can be accessed by index (starts at 0).

- If you want to access Dhoni from collection, index of dhoni paragraph is 0, so you can access as

```
  <script>
  var chennai = document.getElementsByClassName("csk");
  console.log(chennai[0]);
  console.log(chennai[1]);
</script>

Output:
<p class="csk" id="dhoni">I am Dhoni from Chennai</p>
<p class="csk">I am Raina from Chennai</p>
```

- You can also use for loop to access all elements at once

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

Styling DOM Elements in JavaScript

- You can also apply style on HTML elements to change the visual presentation of HTML documents dynamically using JavaScript.
- You can set almost all the styles for the elements like, fonts, colors, margins, borders, background images, text alignment, width and height, position, and so on.

Setting Inline Styles on Elements

- Inline styles are applied directly to the specific HTML element using the style attribute. In JavaScript the `style` property is used to get or set the inline style of an element.
- The following example will set the color and font properties of an element with `id="intro"`.

```
<html lang="en">
<head>
    <title>JS Set Inline Styles Demo</title>
</head>
<body>
    <p id="intro">This is a paragraph.</p>
    <p>This is another paragraph.</p>
</body>
</html>
<script>
    // Selecting element
    var elem = document.getElementById("intro");

    // Appling styles on element
    elem.style.color = "blue";
    elem.style.fontSize = "18px";
    elem.style.fontWeight = "bold";
</script>
```

Output:

This is a paragraph.

This is another paragraph.

Live code : Codepen

querySelector():

- The `querySelector()` method returns the `first` element that matches a CSS selector.

Syntax

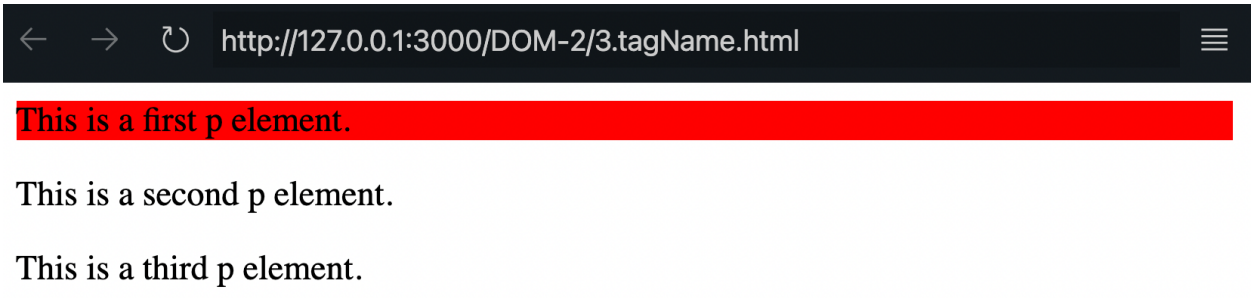
```
document.querySelector(CSS selectors)
```

- Here selectors can be any selector (id,class,tag,universal)

Example

```
<html>
  <body>
    <p>This is a first p element.</p>
    <p>This is a second p element.</p>
    <p>This is a third p element.</p>
  </body>
</html>
<script>
  document.querySelector("p").style.backgroundColor = "red";
</script>
```

Output:

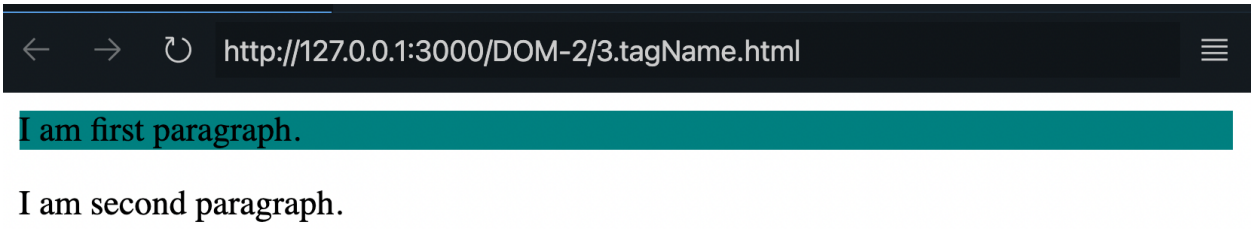


Live code : Codepen

- In the above example, `querySelector` is making background color red to only first element that matches
- Let's take one more example

```
<html>
  <body>
    <p class="example">I am first paragraph.</p>
    <p class="example">I am second paragraph.</p>
  </body>
</html>
<script>
  document.querySelector(".example").style.backgroundColor =
"teal";
</script>
```

Output



Live code : Codepen

- In the above example, `querySelector` is making background color teal to only first element that matches query class example
- We can also use combinators here, but even with combinators it will only return first element

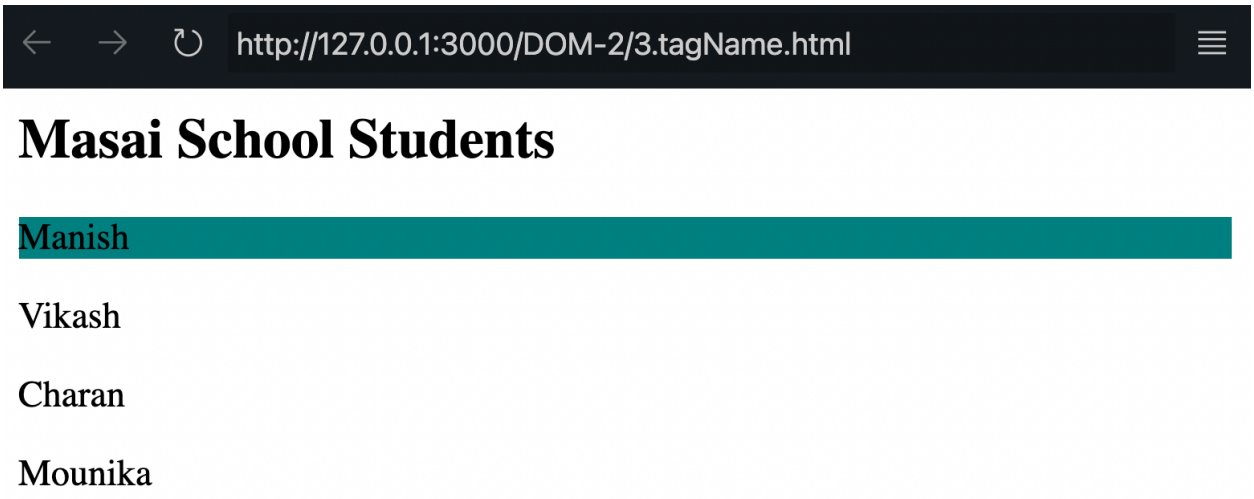
Example:

```
<html>
  <body>
    <h2>Masai School Students</h2>
    <p>Manish</p>
    <p>Vikash</p>
    <p>Charan</p>
    <p>Mounika</p>
```

```
  </body>
</html>
<script>
  document.querySelector("h2~p").style.backgroundColor = "tea
l";
</script>
```

- Basically `h2~p` should return all p's, but `querySelector` will return only first p, in this case Manish

Output:



Live code : Codepen

- To return **all** matches (not only the first), use the `querySelectorAll()` instead.

querySelectorAll():

- The `querySelectorAll()` method returns all elements that matches a CSS selector(s).
- The `querySelectorAll()` method returns a NodeList.

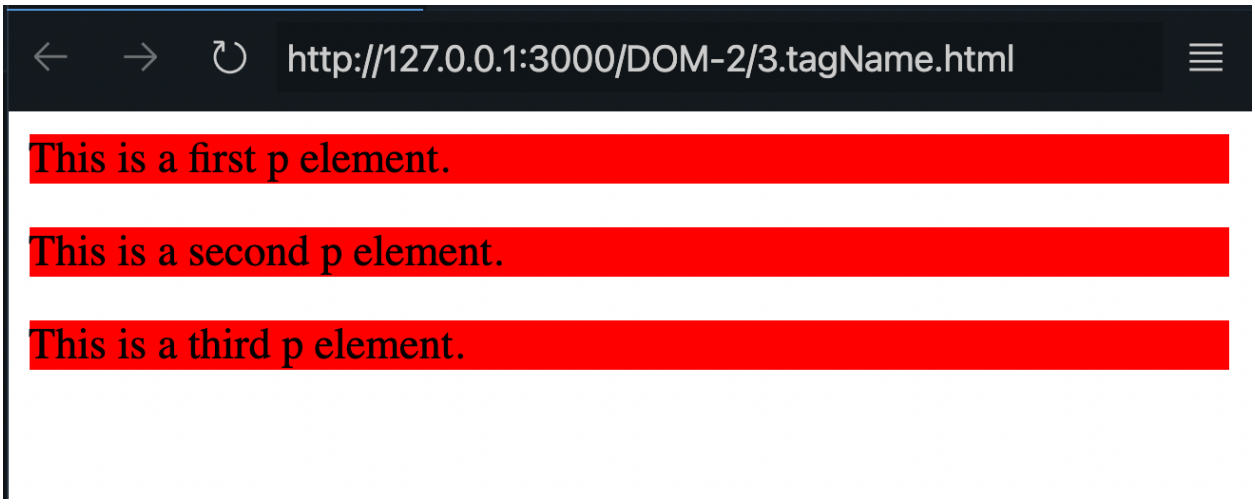
Syntax

```
document.querySelectorAll(CSS selectors)
```

- Here selectors can be any selector (id,class,tag,universal)
- Let’s take the same examples as querySelector

```
<html>
  <body>
    <p>This is a first p element.</p>
    <p>This is a second p element.</p>
    <p>This is a third p element.</p>
  </body>
</html>
<script>
  // This won't work because querySelectorAll() method return
s a NodeList.
  // document.querySelectorAll("p").style.backgroundColor =
"red";
  // So we need to use for loop to select all elements

  var list = document.querySelectorAll("p");
  for (var i = 0; i < list.length; i++) {
    list[i].style.backgroundColor = "red";
  }
</script>
```

Output

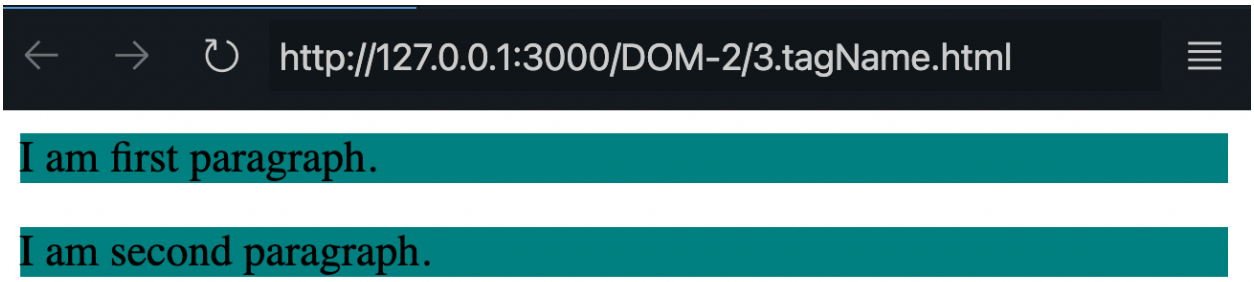


Live code : Codepen

Example 2

```
<html>
  <body>
    <p class="example">I am first paragraph.</p>
    <p class="example">I am second paragraph.</p>
  </body>
</html>
<script>
  var list = document.querySelectorAll(".example");
  for (var i = 0; i < list.length; i++) {
    list[i].style.backgroundColor = "teal";
  }
</script>
```

Output:

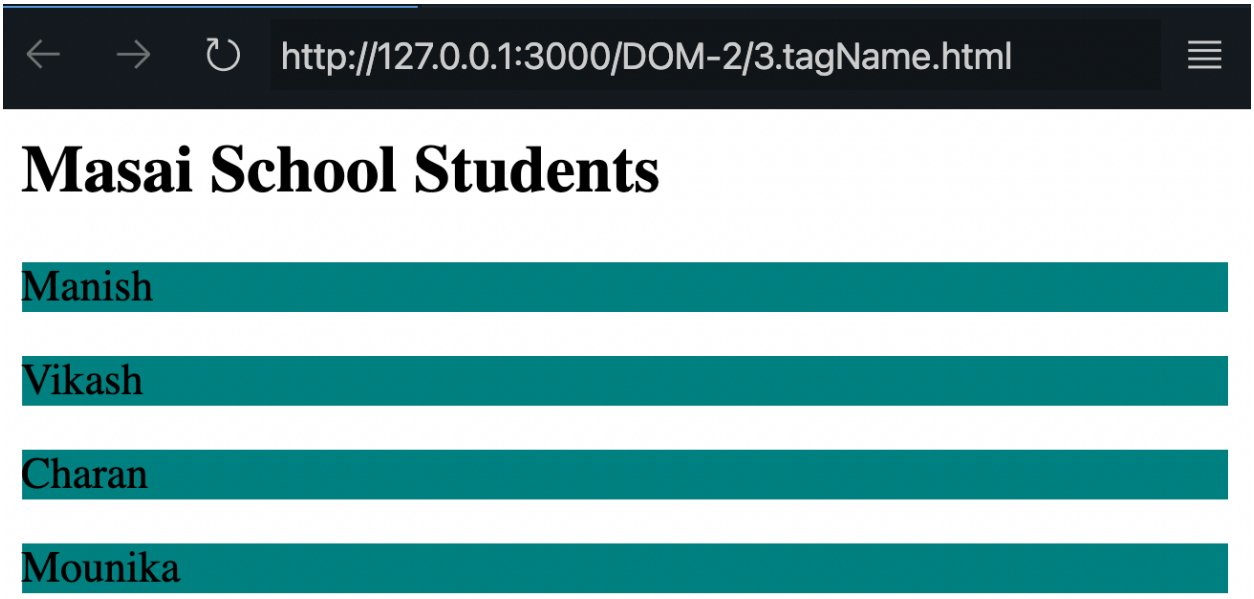


Live code : Codepen

Example 3:

```
<html>
  <body>
    <h2>Masai School Students</h2>
    <p>Manish</p>
    <p>Vikash</p>
    <p>Charan</p>
    <p>Mounika</p>
  </body>
</html>
<script>
    var list = document.querySelectorAll("h2~p");
    for (var i = 0; i < list.length; i++) {
    list[i].style.backgroundColor = "teal";
  }

</script>
```

Output:



Live code : Codepen

EVENTS

- Until now, we are using onClick event listener, but we are using it inline to HTML element.

```
<button onClick="likeMe()">Like👍</button>
```

- But this method of adding event is not recommended, since we are writing JS inside HTML tags.
- So, to improve this, we have a method called as `addEventListener()`
- The `addEventListener()` method attaches an event handler to a document, same functionality as onClick,it's just a syntax change.

addEventListener()

Syntax

```
document.addEventListener(event, function)
```

Parameters

Parameter	Description
event	The event name. Do not use the "on" prefix. Use "click" instead of "onclick".
function	The function to run when the event occurs. When the event occurs, an event object is passed to the function as the first parameter. The type of the event object depends on the specified event. For example, the "click" event belongs to the MouseEvent object.

- So, let's try to rewrite DOM-1 example using `addEventListener`

```
<html>
<head>
      <title>OnClick</title>
</head>
<body>
      <button>Like👍</button>
</body>
</html>
```

- To add `addEventListener` we need to catch the element to which we want to add and then add an event to that element

```
<script>
            document.querySelector("button").addEventListener
("click",likeMe)

      function likeMe() {
```

```
            console.log("Someone liked me")
        }
</script>
```

- Here is a list of some common HTML events:

Event	Description
change	An HTML element has been changed
click	The user clicks an HTML element
mouseover	The user moves the mouse over an HTML element
mouseout	The user moves the mouse away from an HTML element
keydown	The user pushes a keyboard key
load	The browser has finished loading the page

- For more events - Click here

Event Object:

Student Task: Panic (red) and Help (green) button are installed in ATMs. When a customer presses one of these buttons, what information is needed for someone to respond to the event?

(Once the students have responded)

- Target: Which ATM is it
- Type: Whether the red Panic button was pressed or the green Help button

Similarly whenever an event occurs on the page, an event “object” is passed to the function as the first parameter, which has:

- Target: HTML element on which the event was fired. Egs: Button, Text Input, Image
- Type: The type of event fired. Egs: Click, Hover, Scroll

What is event.target in JavaScript?

- `target`, is a property of an event which is a reference to the element upon which the event was fired. Just as 'target' means 'aiming at something', it's used to 'aim' at that particular element.
- This property gives us access to the properties of that element.
- Since the target property has given us access to the element, we could then read some of the properties (which are the attributes) and also display them somewhere else.

Syntax

The `target` property can only be obtained if the event of an `element` is being listened to.

```
element.addEventListener("click",myFunction)

function myFunction(){
```

```
    console.log(event) // //event.target is now accessible
}
```

Importance of event.target

It is necessary to have the `target` property when an event is fired. We can do the following with the `target` property of the event.

- Get the `element` that fired the event.
- Access the properties of the `element`.
- Modify some properties of the `element`, such as the CSS, the attributes, etc.

```
<!DOCTYPE html>
<html>
  <body>
    <p>
      Click on button in this document to find out which elem
ent triggered the
      onclick event.
    </p>

    <button>This is a button</button>

    <p id="demo"></p>

    <script>
      document.querySelector("button").addEventListener("clic
k", myFunction);
      function myFunction() {
        var x = event.target;
        document.getElementById("demo").innerHTML =
          "Triggered by a " + x.tagName + " element";
      }
    </script>
```

```
  </body>
</html>
```

Live code : Codepen