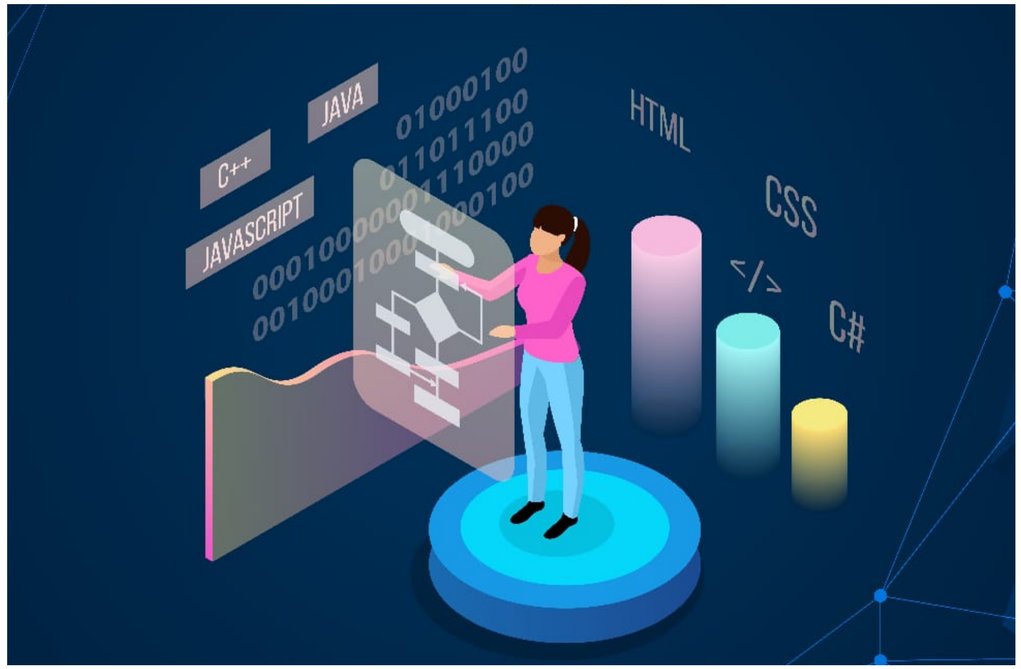


VDC SESSION

VELLORE

PRANAV SINGH

@theBeginner86

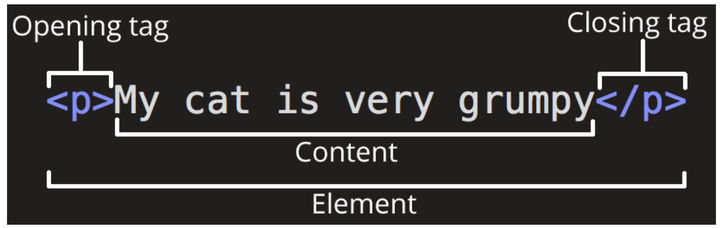
WHAT'S IN TODAY'S MENU?

- HTML
- CSS
- JAVASCRIPT
- Demo - Making simple form with plane HTML and JS

INTRODUCTION TO HTML

CONTENTS

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>HTML</title>
  </head>
  <body>
    ...
    ...
    ...
  </body>
</html>
```

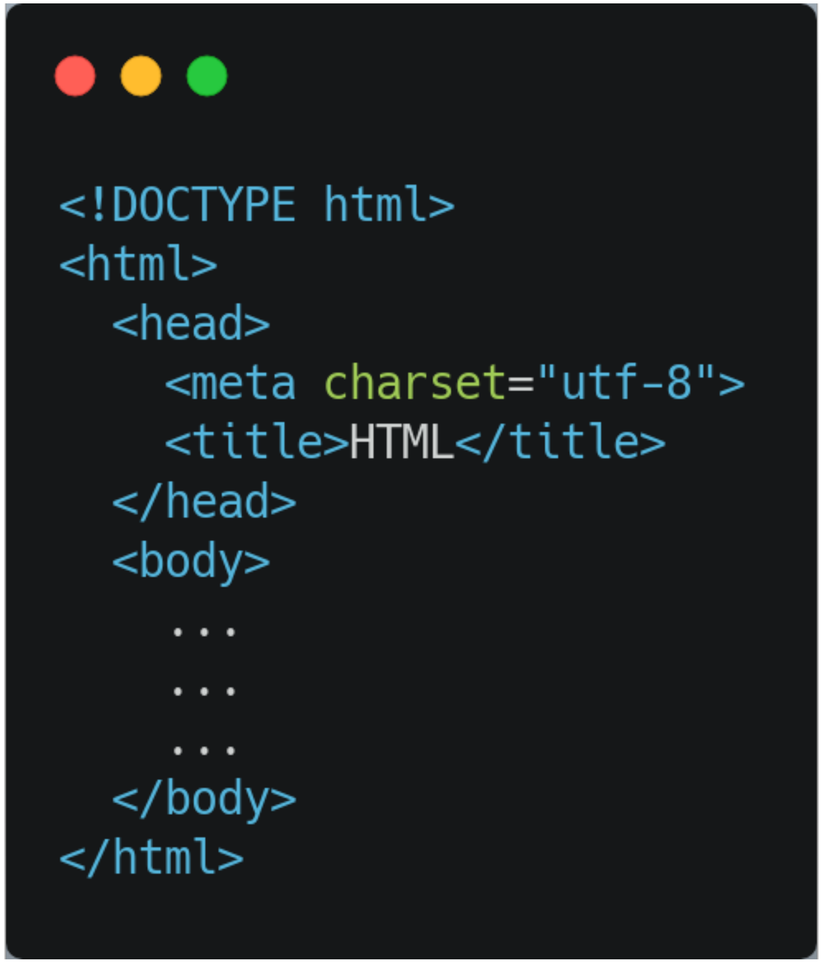


Source: MDN docs

HTML BASICS

- **H**yper **T**ext **M**arkup **L**anguage === HTML
- It helps to structure the website.
- HTML uses various tags.
- HTML consists of series of elements
- These elements are used to define the structure of the website.
- Few tags don't require content hence they are called empty tags and don't have a closing tag. Eg: <img>
- Attributes helps to further enhance the functionality of a tag.

ANATOMY OF HTML DOCUMENT



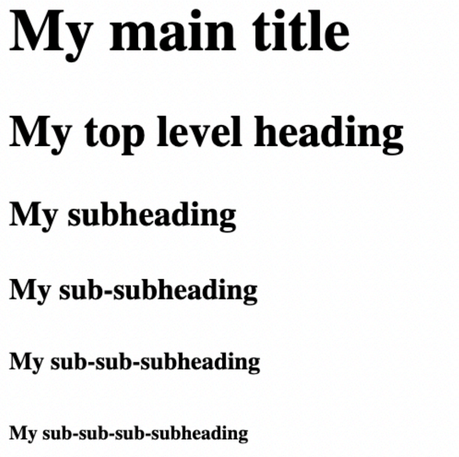
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>HTML</title>
  </head>
  <body>
    ...
    ...
    ...
  </body>
</html>
```

- <!DOCTYPE html> : helps to make the website behaves properly
- <html></html> : main element. Sometimes called root element
- <head></head> : includes content not shown on website but used in various aspects of the website.
- <meta charset-“utf-8”> : sets the character set of the document to UTF-8 which includes character of most languages
- <title></title>: the title of the page visible on the browser tab
- <body></body> : contains all the content that you want to show to web users when they visit your page. Eg: text, images, videos etc

FEW IMPORTANT TAGS WHICH EVERY SHOULD KNOW ABOUT

1. Heading tag

```
<h1>My main title</h1>
<h2>My top level heading</h2>
<h3>My subheading</h3>
<h4>My sub-subheading</h4>
<h5>My sub-sub-subheading</h5>
<h6>My sub-sub-sub-subheading</h6>
```



2. Paragraph tag

```
<p>My first HTML paragrapgh</p>
```

My first HTML paragraph

FEW IMPORTANT TAGS WHICH EVERY SHOULD KNOW ABOUT

3. Unordered List tag

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <ul>
    <li>Item 2.1</li>
    <li>Item 2.2</li>
  </ul>
  <li>Item 3</li>
</ul>
```

- Item 1
- Item 2
    - Item 2.1
    - Item 2.2
- Item 3

4. Ordered List tag

```
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <ul>
    <li>Item 2.1</li>
    <li>Item 2.2</li>
  </ul>
  <li>Item 3</li>
</ol>
```

1. Item 1
2. Item 2
    - Item 2.1
    - Item 2.2
3. Item 3

FEW IMPORTANT TAGS WHICH EVERY SHOULD KNOW ABOUT

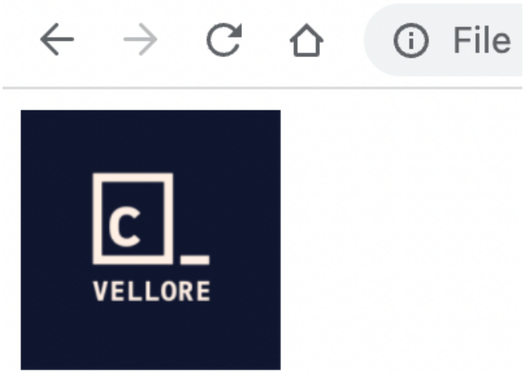
5. anchor tag

```
<a href="https://www.google.com">Click to visit www.google.com</a>
```

Click to visit www.google.com

6. img tag

```
<img src="vdc.png" alt="Vellore Developers Community" width="100" height="100">
```



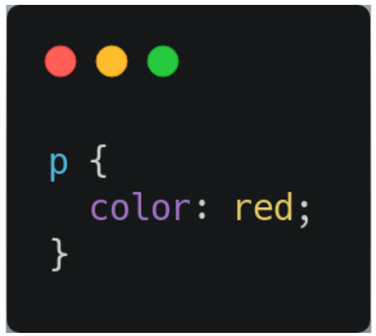
INTRODUCTION TO CSS

CONTENTS

1. Basics of CSS
2. Anatomy of CSS syntax
3. Classes VS Ids
4. How to add CSS?

CSS BASICS

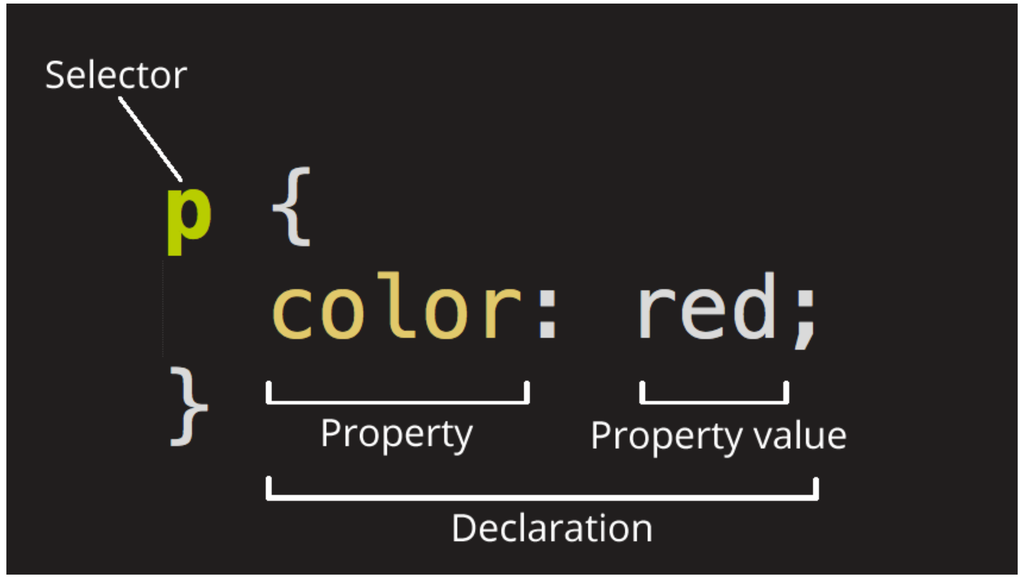
My first HTML paragraph



My first HTML paragraph

- **C**ascading **S**tyle **S**heet === CSS
- Unlike python which is a programming language, or HTML which is a markup language, CSS is a style sheet language.
- Enhances the overlook of the website.
- It styles the web content.
- It helps to selectively style HTML elements.

ANATOMY OF CSS



Source: MDN docs

- Selector: selects particular HTML tag, class, id.
- Declaration: it specifies the property with which you want to style your selectors
- Property: the way you want to style
- Property-value: the value of your property.

CLASSES vs IDS

➤ Classes

1. Used to group related elements that are all going to behave/have same styles

```
<p class="para">Para 1<p>
<p class="para new-para">Para 2<p>
<p class="para">Para 3<p>
```

```
.para {
  color: red;
}

.new-para {
  font-weight: bolder;
}
```

My first HTML paragraph

My first HTML paragraph

My first HTML paragraph

➤ ids

1. These are unique and can be applied to a single HTML tag.

```
<p id="para1">Para 1<p>
<p id="para2">Para 2<p>
<p id="para3">Para 3<p>
```

```
#para1 {
  color: red;
}
#para2 {
  color: blue;
}
#para3 {
  color: green;
}
```

My first HTML paragraph

My first HTML paragraph

My first HTML paragraph

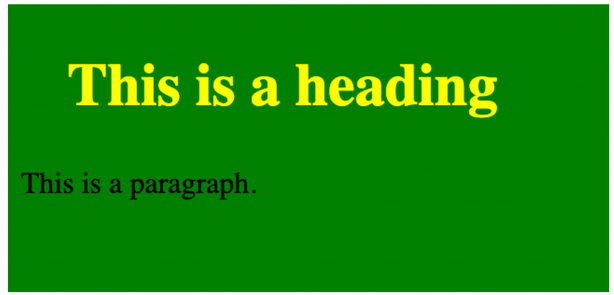
HOW TO ADD CSS

1. External

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```css
body {
  background-color: green;
}
h1 {
  color: yellow;
  margin: 25px;
}
```

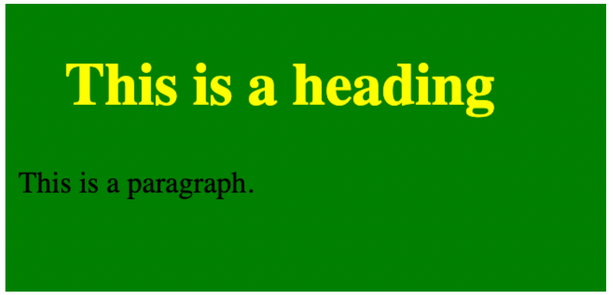
mystyles.css



HOW TO ADD CSS

2. Internal

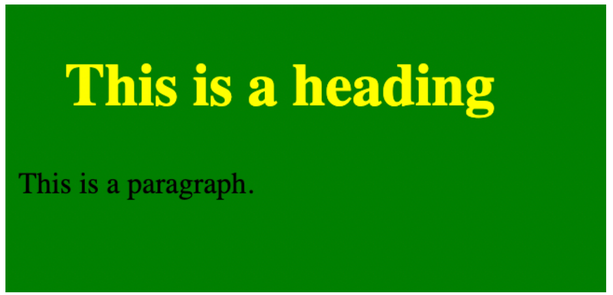
```html
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: green;
}
h1 {
  color: yellow;
  margin: 25px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



HOW TO ADD CSS

3. Inline

```html
<!DOCTYPE html>
<html>
<head>
  <title>css</title>
</head>
<body style="background-color: green;">

<h1 style="color: yellow;margin: 25px;">This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



INTRODUCTION TO JS

Netscape

JOURNEY FROM "MOCHA" TO JAVASCRIPT

Marc Andreessen

Brendan Eich

- Popularity of Google Chrome today, is same as that of Netscape navigator in mid 1990s.
- Netscape ***VS*** Microsoft
- Netscape wanted the code to run on the browser (or client side) itself.
- Brendan developed the first release in just 10 days.
- In the first release it was called, LiveScript.
- Then it changed to JScript but ECMA chose to standardise it with the name ECMAScript.
- Due to the popularity of JAVA they named it JAVASCRIPT.

DIFFERENCE BETWEEN JAVA AND JAVASCRIPT

- Java is a complied language while Javascript is an interpreted language.
- Java has static type checking while Javascript has dynamic type checking.
- Running Java application requires JVM or browser while Javascript runs on browser.
- Java is primarily used for backend while Javascript can be used for both backend and frontend.

Theses languages are as much similar as **Car** and **Carpet** are!!

CONCEPTS OF JAVASCRIPT

CONTENTS

1. DECLARING VARIABLES IN JS

➤ Keywords are used:

1. var
2. let (ES6) - released on 2015
3. const (ES6) - released on 2015

➤ Declaration allowed: **const**, **var**, **let**

```
var a = 1
var b = "string"
var c = true
```

```
let a = 1
let b = "string"
let c = true
```

```
const a = 1
const b = "string"
const c = true
```

➤ Redeclaration: **var**

```
var a = 1

function myfunction(){
  ...
  ...
}

var a = 2
```

```
let a = 1

function myfunction(){
  ...
  ...
}

let a = 2 //error
```

```
const a = 1

function myfunction(){
  ...
  ...
}

const a = 2 //error
```

➤ Updation: **let**, **var**

```
var a = 1
function myfunction(){
  ...
  ...
}
a = 2
```

```
let a = "hello"
function myfunction(){
  ...
  ...
}
a = "world"
```

```
const a = 1
function myfunction(){
  ...
  ...
}
a = 2 //error
```

2. ARRAYS

- Arrays in JS are referred to as a list which contains a collection of data which may or may not be same.

```
var myList = [1, 2, "VDC Session", true, false]

myList.length //5

myList[0] //1
myList[1] //2
myList[2] //"VDC Session"
myList[3] //true
myList[4] //false
```

```
var myList = []

myList.push("obj1")
myList.push("obj2")
myList.push("obj3")


myList[0] //"obj1"
myList[1] //"obj2"
myList[2] //"obj3"

myList.length //3

myList.pop()

myList.length //2
```

```
var myString = 'VDC'

myString.length //3

myString[0] //'V'
myString[1] //'D'
myString[2] //'C'

myString.slice(0, 2) //'VD'
myString.slice(-1) //'C'
myString.slice(-2) //'DC'
myString.slice(1) //'DC'
```

3. OBJECTS AND DESTRUCTURING

- An object is a collection of properties, and a property is an association between a name (or key) and a value.

```
var myObj = {
  key1: 1,
  key2: "friday",
  key3: true
  ...
  ...
}

var a = myObj.key1 // var a = 1
var b = myObj.key2 // var b = "friday"
var c = myObj.key3 // var c = true
```

```
var myObj = {
  key1: 1,
  key2: "friday",
  key3: true
  ...
  ...
}

var {key1, key2} = myObj //key1 = 1
                         //key2 = "friday"
```

```
var myObj = {
  key1: 1,
  key2: "friday",
  key3: true
  ...
  ...
}

var {key1: newKey1, key2: newKey2} = myObj //newKey1 = 1
                                           //newKey2 = "friday"
```

4. FUNCTIONS

- Basic syntax

```
function myFunction(para1, para2, para3, ...){
  ...
  ...
  ...
}

myFunction(para1, para2, para3, ...)
```

```
function myFunction(para1, para2, para3, ...){
  ...
  ...
  ...
  return para1 + para2 + para3 ...
}

var sum = myFunction(para1, para2, para3, ...)
```

- Anonymous Functions

```
var sum = function (num1, num2) {
    return num1 + num2
}

sum(1, 2)
```

- Arrow Functions

```
var sum = (num1, num2) => {
    return num1 + num2
}

sum(1, 2)
```

```
var a = 1
var b = 1
function (){
  return 1 + a + b
}

() => 1 + a + b
```

```
function (a){
  return 1 + a
}

a => 1 + a
```

```
function (a, b){
  return a + b
}

(a, b) => a + b
```

- Callback functions

```
function sum(sumFunction, num1, num2){
    return sumFunction(num1, num2)
}

function sumFunction(num1, num2){
    return num1 + num2
}

var a = 1
var b = 2
var c = sum(sumFunction, a, b)
```

LOOPS

- For

```
for ([initialExpression]; [conditionExpression]; [incrementExpression]){
  ...
  ...
  ...
}
```

- do..while

```
do {
  ...
  ...
  ...
} while (condition);
```

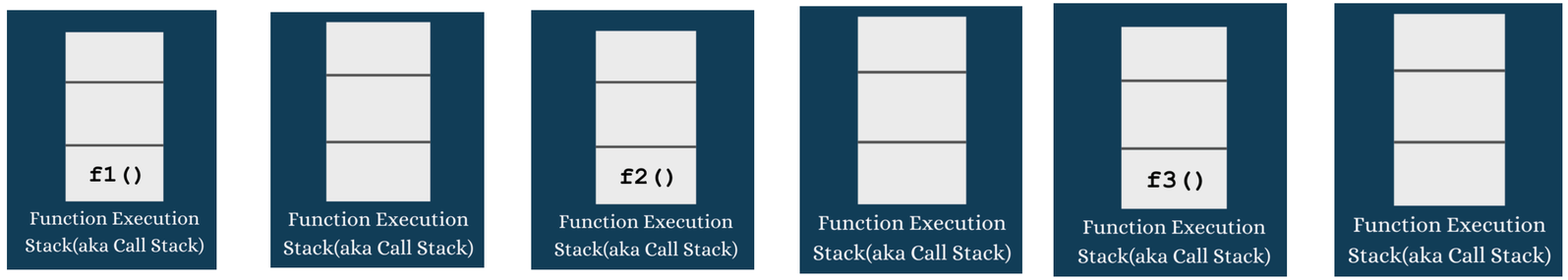
- while

```
while(condition){
  ...
  ...
  ...
}
```

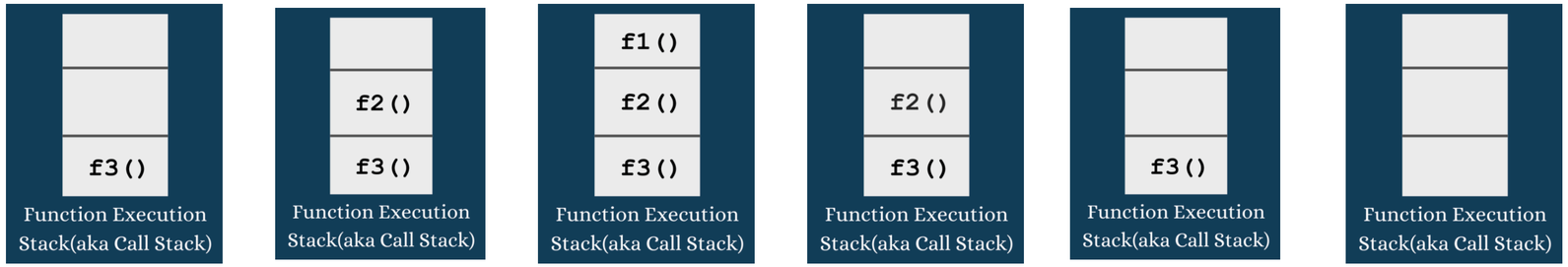
5. ASYNCHRONOUS AND SYNCHRONOUS

- Synchronous: When your code works in a sequence – each statement waits for the previous statement to finish before executing.
- Function Execution stack: The JavaScript engine maintains a **stack** data structure called **function execution stack**. The purpose of the stack is to **track the current function in execution**. An event loop tracks all the functions in function execution stack.

```
function f1() {
  // some code
}
function f2() {
  // some code
}
function f3() {
  // some code
}

// Invoke the functions one by one
f1();
f2();
f3();
```




```
function f1() {
  // Some code
}
function f2() {
  f1();
}
function f3() {
  f2();
}
f3();
```



- Asynchronous: The word asynchronous means not occurring at the same time.

Browser APIs/Web APIs: Way to interact between different services/machines.

Callback Queues/Task Queues: keeps track of all the callback functions

```
function myFunction() {
  console.log('print me');
}

setTimeout(myFunction, 2000);
```

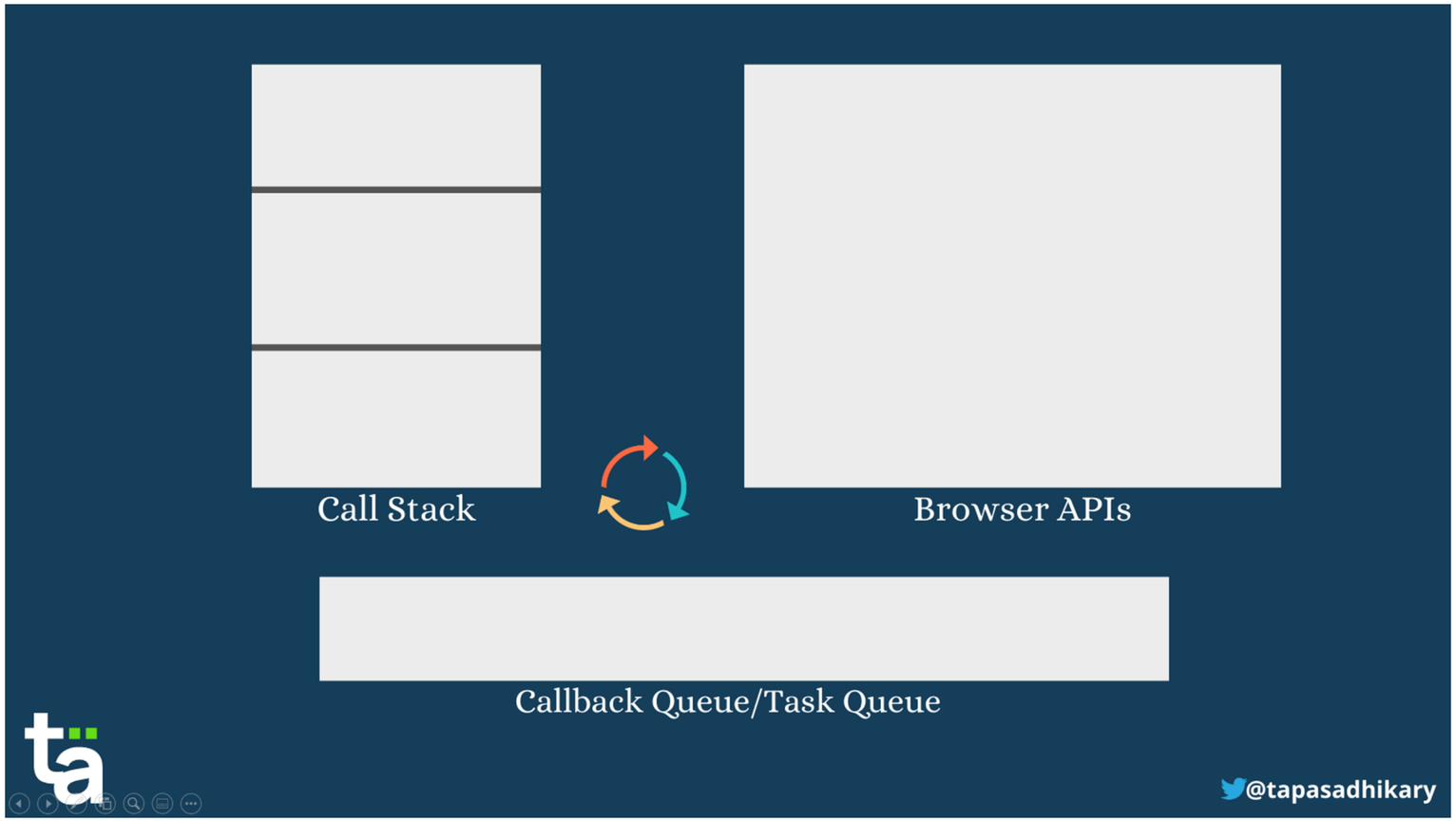
```
function test2() {
  console.log('test2');
}

function test() {
  console.log('test');
}

setTimeout(test2, 2000);
test();
```

```
function test2() {
  console.log('test2');
}

function test() {
  console.log('test');
}

setTimeout(test2, 2000);
test();

//test
//test2
```

```
function f1() {
    console.log('f1');
}

function f2() {
    console.log('f2');
}

function main() {
    console.log('main');

    setTimeout(f1, 0);

    f2();
}

main();
```

```
function f1() {
    console.log('f1');
}

function f2() {
    console.log('f2');
}

function main() {
    console.log('main');

    setTimeout(f1, 0);

    f2();
}

main();

//main
//f2
//f1
```

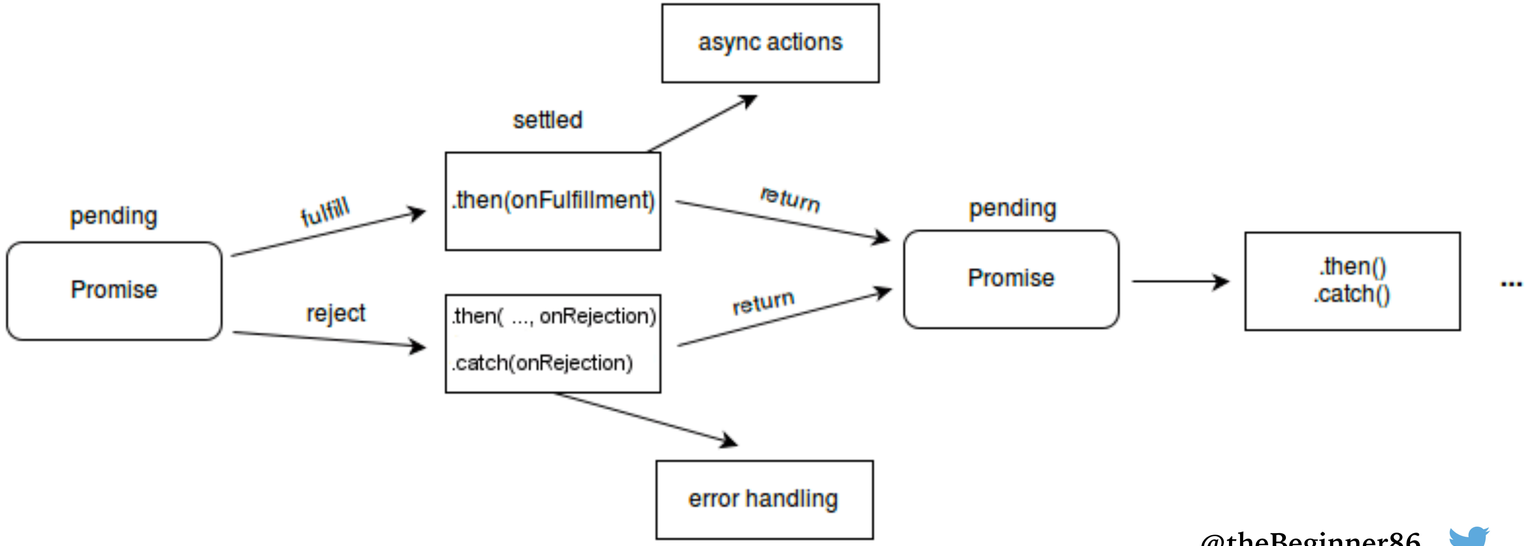
➤ Working of Callback functions:



- Promises: These are special objects that are related to asynchronous operations. That is, **asynchronous operation/methods** returns a *promise* when we are expecting to get some value back from that function.
- Job queue: Every time a promise occurs in the code, the executor function gets into the job queue. And the **event loop** gives **job queue a higher priority** than **function execution stack**.
- States of a promise:

1. pending: initial state, neither fulfilled nor rejected.
2. fulfilled: meaning that the operation was completed successfully.
3. rejected: meaning that the operation failed.

```
const promise = new Promise((resolve, reject) =>
        resolve('I am a resolved promise');
);

promise.then(result => console.log(result))
```

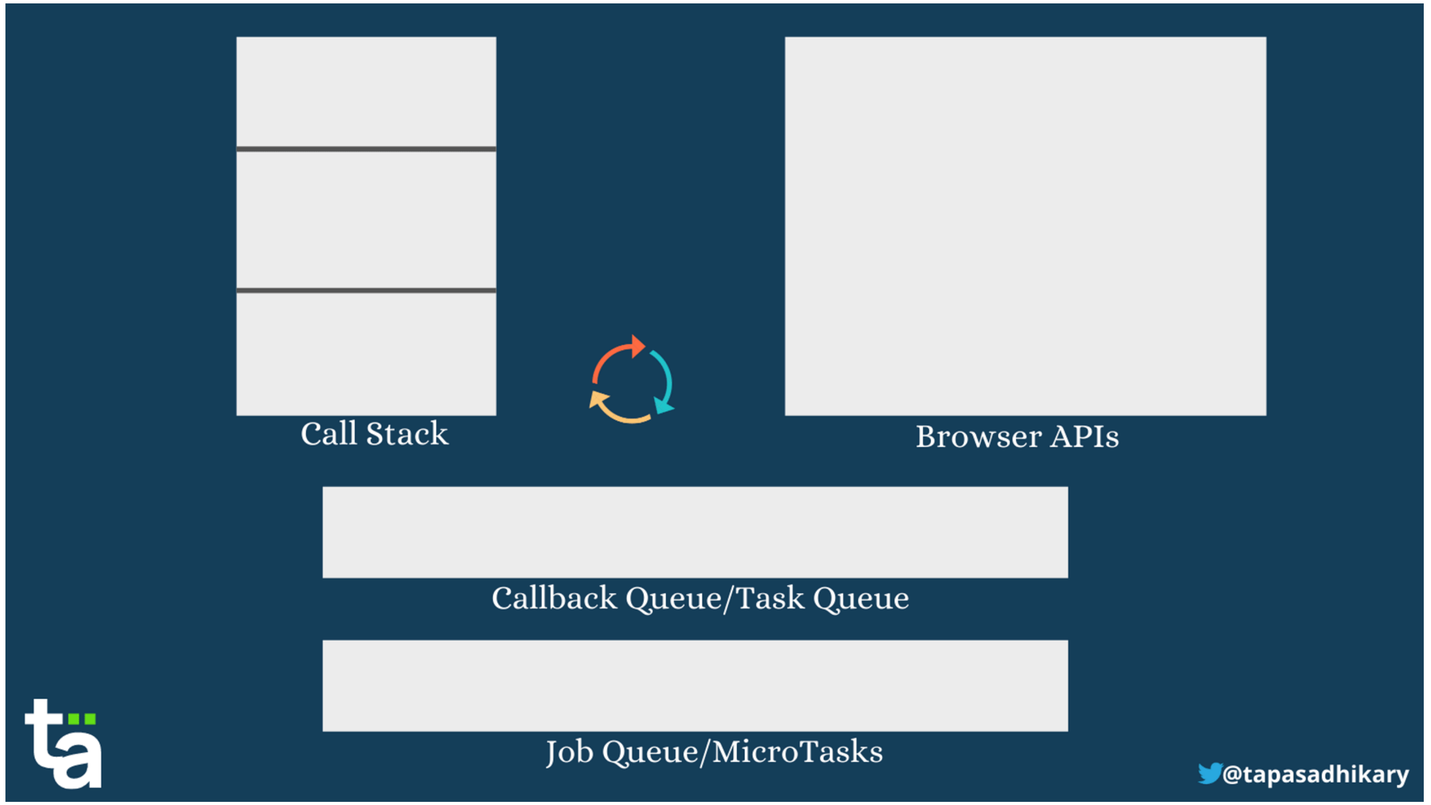


➤ Working of promises:

```
function f1() {
    console.log('f1');
}

function f2() {
    console.log('f2');
}

function main() {
    console.log('main');

    setTimeout(f1, 0);

    new Promise((resolve, reject)
=>
        resolve('I am a promise')
    ).then(resolve =>
console.log(resolve))

    f2();
}

main();
```

```
function f1() {
    console.log('f1');
}

function f2() {
    console.log('f2');
}

function main() {
    console.log('main');

    setTimeout(f1, 0);

    new Promise((resolve, reject)
=>
        resolve('I am a promise')
    ).then(resolve =>
console.log(resolve))

    f2();
}

main();

//main
//f2
//I am a promise
//f1
```

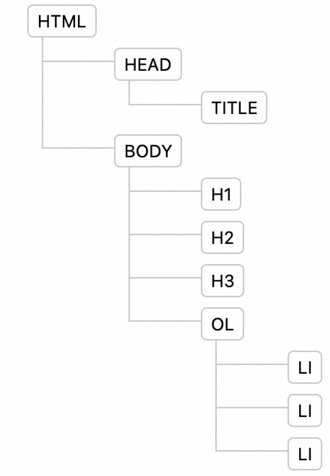
➤ Working of job queue:



5. DOM (DOCUMENT OBJECT MODEL) AND JQUERY

- The Document Object Model (DOM) is the data representation of the objects that comprise the structure and content of a document on the web.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <h1>VDC Session</h1>
    <h2>Friday</h2>
    <h3>By Pranav Singh</h3>
    <ol>
        <li>item1</li>
        <li>item2</li>
        <li>item3</li>
    </ol>
</body>
</html>
```



- jQuery: It is a Javascript library that helps in simplifying DOM manipulations.

```
document.querySelector("p")
$("p")

document.getElementById("first-h1")
$("#first-h1")

document.getElementsByClassName("second-h1")[0]
$(".second-h1")
```

```
document.getElementById("first-h1").style.color = "red"
$("#first-h1").style.color = "red"

document.getElementsByClassName("second-h1")[0].style.color =
"red"
$(".second-h1").style.color = "red"
```

John Resig

- How to add jQuery in our website?

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
```

```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

6. MISCELLANEOUS

- map()

```
var arr = [1, 2, 3, 4]
var arr1 = arr.map(x => x+1) //arr1 = [2, 3, 4, 5]
```

- console.log()

```
var arr = [1, 2, 3, 4]
console.log(arr) // [1, 2, 3, 4]

var a = 1
console.log(a) // 1

console.log('hello word') //'hello world'
```

- Formatted string

```
for(var i=0; i<3; i++){
  console.log(`i=${i}`)
}

//i=0
//i=1
//i=2
```

- Asynchronous function

```
async function myfunction(){
  ...
  ...
}

myfunction()
```

- await keyword

```
async function myfunction(){
  var returnedObj = await api.fetch()
  ...
  ...
}

myfunction()
```

- try..catch()

```
try {
  async function myfunction(){
  var returnedObj = await api.fetch()
  ...
  ...
  }
} catch(err){
  console.log(err)
}

myfunction()
```

DEMO TIME

In this demo, we would be making a simple form using plane JS and HTML