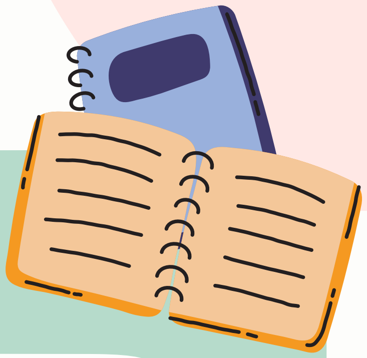
JAVASCRIPT NOTES

JAVASCRIPT (JS) IS A LIGHTWEIGHT, INTERPRETED, OR JUST-IN-TIME COMPILED PROGRAMMING LANGUAGE . JAVASCRIPT IS A PROTOTYPE-BASED, MULTI-PARADIGM, SINGLE-THREADED, DYNAMIC LANGUAGE, SUPPORTING OBJECT-ORIENTED, IMPERATIVE, AND DECLARATIVE (E.G. FUNCTIONAL PROGRAMMING) STYLES. READ MORE ABOUT JAVASCRIPT.



DATA TYPE IN JS

Primitive

there are six
primitive data type

- undefined
- number
- boolean
- string
- BigInt
- symbol

Non -Primitive

LOOPS IN JAVASCRIPT

for loop
While loop
do while loop
for in
for of
for each

For in loop

The JavaScript for in statement loops through the properties of an Object:

```
for (key in object) {
 // code block to be
executed
}
```

For of loop

The JavaScript for of statement loops through the values of an iterable object.

```
for (variable of
iterable) {
// code block to be
executed
}
```

FUNCTION

A JavaScript function is a block of code designed to perform a particular task.

```
// Function to compute the product
of p1 and p2

//function is defined
function myFunction(p1, p2) {
return p1 * p2;
}

myFunction() ; //function is called
```

FUNCTION PARAMETER VS FUNCTION ARGUMENT

- Fuction paramter are the name listed in the function's defenation
- Function argument are the real value passed to the function

FUNCTION EXPRESSION

- Create a function and pass it into a variable

ANONOYMOUS FUNCTION

Anonymous Function is a function that does not have any name associated with it.

```
function() {
// Function Body
}
```

How JS work ?

ANY JS CODE IS WORKED IN THE FOLLWING WAYS .

1. CODE COMPILATION
2. CODE EXECUTION

FIRSTLY , THE CODE IS COMPLIED TO CHECK THE ERROR AND DETERMINE THE APPROCIATE SCOPE OF THE VARIABLE.

IT HAS THREE PHASE

1. LEXING
2. PARSING
3. CODE GENERATION

SECONDLY THE CODE IS EXCUTED INSIDE THE EXCUTION CONTEXT

EXECUTION CONTEXT

EXECUTION CONTEXT IS THE CONCEPT FOR DESCRIBING THE INTERNAL WORKING OF A CODE. IN JAVASCRIPT, THE ENVIRONMENT THAT ENABLES THE JAVASCRIPT CODE TO GET EXECUTED IS WHAT WE CALL JAVASCRIPT EXECUTION CONTEXT. DURING THE EXECUTION CONTEXT, THE SPECIFIC CODE GETS PARSED LINE BY LINE THEN THE VARIABLES AND FUNCTIONS ARE STORED IN THE MEMORY. AN EXECUTION CONTEXT IS SIMILAR TO A CONTAINER THAT STORES VARIABLES, AND THE CODE GETS EVALUATED AND THEN EXECUTED. THUS, IT IS THE EXECUTION CONTEXT THAT PROVIDES AN ENVIRONMENT FOR THE SPECIFIC CODE TO GET EXECUTED.

TYPE OF EXECUTION CONTEXT

GLOBAL EXECUTION CONTEXT

GEC / Global Execution Context is also called the base/default execution. Any JavaScript code which does not reside in any function will be present in the global execution context.

FUNCTIONAL EXECUTION CONTEXT

It is a of context which is created by the JavaScript engine when any function call is found. Every function has its own execution context, and thus unlike GEC, the FEC can be more than one. Also, FEC can access the entire code of the GEC, but it is not possible for GEC to access all the code of the FEC.

EVAL EXECUTION CONTEXT

Any JS code that gets executed within the eval function creates and holds its own execution context. However, the eval function is not used by the JavaScript developers, but it is a part of the Execution Context.

SOme important term in js

TEMPORAL DEAD ZONE (TDZ)- the area of a block where a variable is inaccessible until the moment the computer completely initializes it with a value.

CALL STACK -A data structure that works behind the scenes to support the method call/return mechanism

CLOSURE- the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment).

Argument - It's array like object which store the parameter passed in a function when it's called .

OBJECT IN JS

In JavaScript, an object is a standalone entity, with properties and type.

Example -

```
const person={
name:"sunny,
age:21,
hobbies:["vlogging", "football",
"travelling"]
}.
```

Acessing the object element

objectName.keyName

or

objectName["keyName"]

THIS KEYWORD

- In JavaScript, the this keyword refers to an object.
- Which object depends on how this is being invoked (used or called).
- The this keyword refers to different objects depending on how it is used:

1. In an object method, this refers to the object.
2. Alone, this refers to the global object.
3. In a function, this refers to the global object.
4. In a function, in strict mode, this is undefined.
5. In an event, this refers to the element that received the event.

EVENT

Events are actions or occurrences that happen in the system you are programming, which the system tells you about so your code can react to them. For example, if the user clicks a button on a webpage, you might want to react to that action by displaying an information box.

Few example of events -

onclick()

onkeypress()

onhover()

STRING

JavaScript strings are for storing and manipulating text.

Some important function

- length()
- slice()
- charAt()
- trim()
- split()
- concat()

ARRAY

An array is a special variable, which can hold more than one value:

- Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays.
- Arrays use numbers to access its "elements"

SOme important method of array

- Map()
- Filter()
- Reduce()
- sort()
- find()
- every()
- some()
- fill()
- splice()

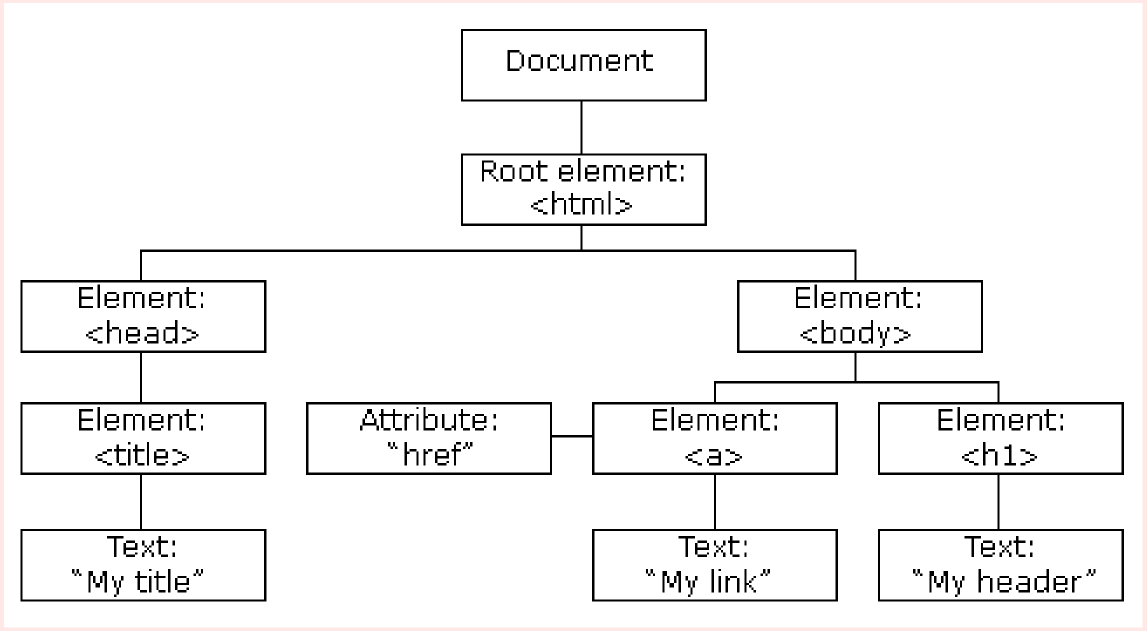
DOM

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elemen

When a web page is loaded, the browser creates a Document Object Model of the page. The HTML DOM model is constructed as a tree of Objects:

DOM TREE



DOM METHODS

getElementById(id)	Find an element by element id
getElementsByTagName (name)	Find elements by tag name
getElementsByClassNa me(name)	Find elements by class name
querySelector("id/cla ss)	The querySelector() method returns the first element that matches a CSS selector
querySelectorAll("id/c lass)	The querySelector() method returns all the element that matches a CSS selector

DIFFRENCE BETWEEN GETELEMENTBYID AND QUERYSELECTOR

- getElementById

Syntax: Const element = document.getElementById();

Return a refrence to the element by it's ID.

If the specified ID is not found then it will return null.

- querySelector

Syntax: Const element = document.querySelector();

Return the first element with in the document that match the specified group of selector , or null if no match is found .

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

Both querySelector() and querySelectorAll() throw a SYNTAX_ERR exception if the selector(s) is invalid.

ADVANCED JAVASCRIPT

Event Propogation

It determine in which order our element recieve the event or Event propagation is a way to describe the “stack” of events that are fired in a web browser.

Event Bubling

With Event Bubbling, the event is first captured and handled by the innermost element and then propagated to outer elements.

Syntax: id.addEventListener('event', function,false

Event Capturing

With Event Capturing, the event is first captured by the outermost element and propagated to the inner elements.
Capturing is also called "trickling", which helps remember the propagation order

Syntax: id.addEventListener('event', function,true)

EVENT PROPOGATION

Event propogation can be characterised into 3 categories

Capture phase

Going form the window to the event target phase

Target phase

It's the target phase

Bubble phase

From the event target parent back to the window

HIGHER ORDER FUNCTION

Function which take another function as an argument is known as higher order function

CALL BACK FUNCTION

Function which get passed as an argument to another function is called call back function

let's see an example

```
//array of names to be used in the function
const names= ['John', 'Tina','Kale','Max']
//Function using function fn as a parameterfunction
useFunction(arr,fn){for(let i=0; i<arr.length; i++){
fn(arr[i]);}}
//Function that is being used as a parameter
function argFn (name){
console.log("Hello " + name );}
//calling useFunction() with argFn() as a parameter
useFunction(names,argFn);
```

ASYNCHRONOUS JAVACRIPT

HOISTING

The process whereby the interpreter appears to move the declaration of functions, variables or classes to the top of their scope, prior to execution of the code.

EXAMPLE-

```
var x; // Declare x
x = 5; // Assign 5 to x
```

SCOPE CHAINING AND LEXIAL SCOPE

- The scope chain is used to resolve the values of variable name in js
- Scope chain in js is lexially defined , which mean that we can see what the scope chain will be by looking at the code.
- Lexical scope mean , the inner function can get acess to thier parent function variable but the vice versa of this in not true .

CLOSURE

A Closure is a combination of function bundle together with refrence to its surronding state(the lexical environment).
In other words , a closure give you acess to an outer function's scope from a inner function .

ASYNCHRONOUS JAVACRIPT

SET TIME OUT

Excute a function after waiting for a specfied no of milisecond

SYNTAX

⟶ setTimeout(function,milisecond)

EXAMPLE

```
console.log("script start");
const id = setTimeout(() => {
console.log("inside setTimeout");
}, 1000);
for (let i = 1; i < 100; i++) {
console.log("....");
}
console.log("settimeout id is ", id);
console.log("Script end");
```

ASYNCHRONOUS JAVACRIPT

SET INTERVAL

same as setimeout but repeat the function continuoesly

SYNTAX

$\longrightarrow$ setInterval(function,milisecond)

EXAMPLE

```
console.log("script start");
setInterval(() => {
console.log(total);
console.log(Math.random());
}, 500);
console.log("script end");
```

ASYNCHRONOUS JAVACRIPT

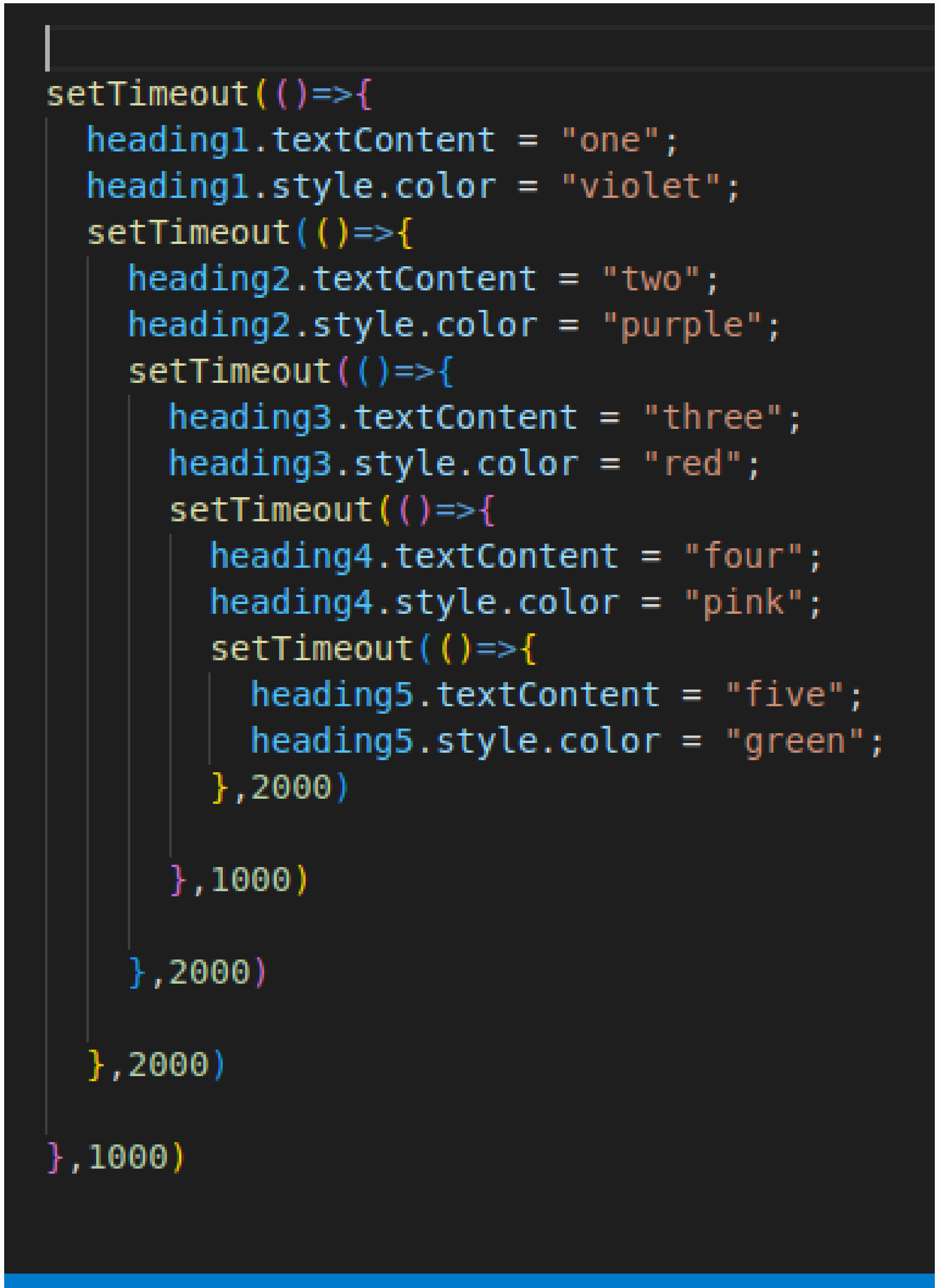
FUNCTON CURRYING

Currying is a technique of evaluating function with multilple argument , into sequence of function with single argument.

EXAMPLE : Sum(5) (6) (7)

```
function Sum(num 1){
console.log(num 1); //5 will be printed
return function(num2){
console.log( num 1, num 2); //5 and 6 will be
printed
return function (num 3){
console.log(num 1, num 2, num 3);
// , 6,7 will be printed
}
}
}
```

Call Back Hell



```
setTimeout(()=>{
  heading1.textContent = "one";
  heading1.style.color = "violet";
  setTimeout(()=>{
    heading2.textContent = "two";
    heading2.style.color = "purple";
    setTimeout(()=>{
      heading3.textContent = "three";
      heading3.style.color = "red";
      setTimeout(()=>{
        heading4.textContent = "four";
        heading4.style.color = "pink";
        setTimeout(()=>{
          heading5.textContent = "five";
          heading5.style.color = "green";
        },2000)

      },1000)

    },2000)

  },2000)

},1000)
```

XMLHttp Request

XMLHttpRequest (XHR) objects are used to interact with servers. You can retrieve data from a URL without having to do a full page refresh. This enables a Web page to update just part of a page without disrupting what the user is doing. XMLHttpRequest is used heavily in AJAX programming.