Javascript

- Js uses what's referred to an event-driven model of execution,

  i.e, when you embed Js code in a web page, it isn't run until the event associated with it is triggered.

- Weakly typed lang (Implicit conversion) event like → loading or leaving the page without user Interference
- It runs inside browser, that has direct access to all elements in a web document.
- Structure of js-script.
- You can put <script> tag either in <head> or in <body>

→ In <head> tag it ensure that scripts are loaded before the content of page,

- It's beneficial for defining variable or initializing libraries, but slow down the speed

→ In <Body> tag → content load first (Improving Speed)

useful for scripts, that are non-essential or manipulate page content, after it has uploaded.

- Comments are done using → double forward slash (//) (multi-line comment) or /* ... */
- External js

<script src="script.js">

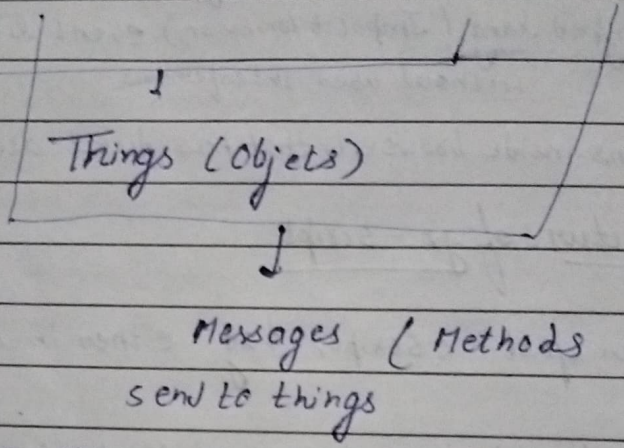
- <noscript>
  ↳ It works when js is disabled or doesn't supported by browser.

eg <noscript>
Your Js is disabled
</noscript>

- The console we use here, is also known as REPL (Read-evaluate-Print loop)

- The javascript language

(a) document.write(') → we pass arguments here (b) window.alert()



Messages (Methods) send to things

↓ It accepts string as argument, but if we pass Integer or any other, it doesn't throw error, it will convert it and prints out.

- So js is case-sensitive, but its not strict about data types

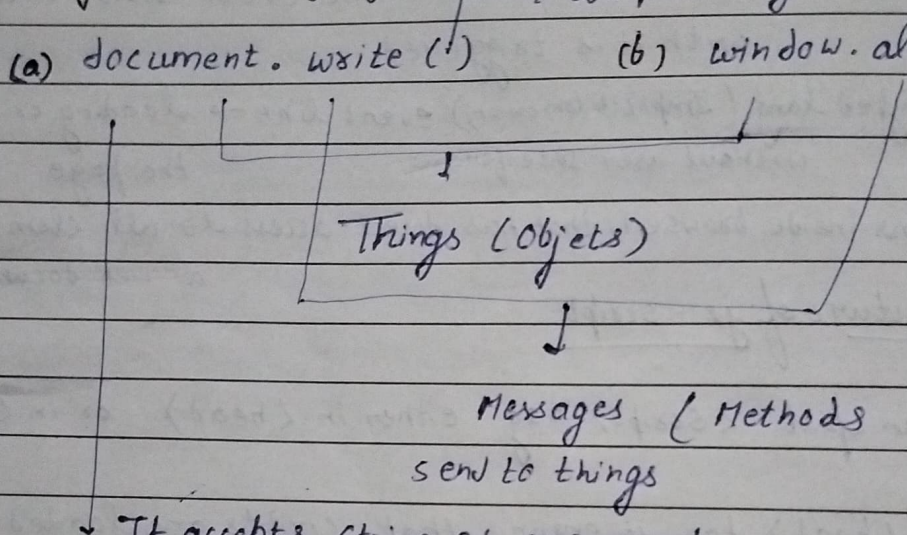
- It has also expression or operator, same as python,

extra :→ Math.sqrt(25)
:) 5

- ~~Semicolon~~ Semicolon rule (;) to indicate end of statement
  ↳ use (;) after each command, except, the last one.
  ↓ (because new-line serves same purpose as semi-colon)

- The console we use here, is also known as REPL (Read-evaluate-Print loop)

- The javascript language



↓ It accepts string as argument, but if we pass Integer or any other, it doesn't throw error, it will convert it and prints out.

- So js is case-sensitive, but its not strict about data types

- It has also expression or operator, same as python,

extra :→ Math.sqrt(25)
=> 5

- ~~Semicolon~~ Semicolon rule (;) to indicate end of statement
  ↳ use (;) after each command, except, the last one.
  ↓ (because new-line serves same purpose as semi-colon)

- **Variables**

```
> var firstname = "Ram";
> var lastname = "Sharma";
> var name = firstname + " " + lastname;
> using a quote within a string → var greeting = "\"Hello!\" is a greeting"
> name
Ram Sharma
```

- Variable name contain only letter, numbers, and undersore (_) or dollar ($) character
- can't start with number.
- can't use reserve words.
- Style wise → use this : my-variable
  or that : myVariable

- also define variable, without assigning value to it,

```
var my-variable;
```

after that

```
my-variable = 5
```

Control Structure

~~Loops~~

```
if ( ? ) { ? ;}
```

Condition Statement — Statement to be executed if condition is true

```
else { ? ;}
```

If condition is false this will be executed.

also, if ( ) { }
else if ( ) { } → elif
else { }
↳ In one statement we skip this

Page No.
Date

e.g var color = "red";
if (color == "blue") { color = "red"; }
else { color = "blue"; }
color;

{ negator, !(1==2) o/p: True }

Loops

(i) for loops

for (variable_declaration ; Boolean expression ; Increment/decrement)
which terminate loop, when it's false
{ console.log ("Iteration number" + count); }
↑ The statement that you want to print

e.g for (var count = 1; count <= 10; count++) {
console.log ("Iteration number" + count); }

(ii) while loops

eg: var color = "blue"; → Condition
while (color == 'blue') {
console.log ("Color is blue.");
if (Math.random() > 0.5) { color = 'not blue'; } }

(0,1)

stopping condition

- Caution for old or non-standard browser,

↳ use, HTML comment tag (<!-- and -->), in case of non-supporting (old.) js browser,

e.g
```
<script type="text/javascript">
<!-- (Your J.S code)
//-->
</script>
```

Here '//' is use → in case brower support J.S, the closing html tag, will treat as comment, and then, <js script> will perfectly run.

- Getting debugging J.S errors (i.e, How to access j.s error messages)

↳ use this on Chrome, edge, opera, firefox ⇒ ctrl + shift + J (PC)

⇒ cmd + shift + J (MAC)

- Javascript arrays:

Toys = [['A', 'B', 'C'], ['1', '2', '3'], ['x', 'y', 'z']]

To access → 'x'

document.write(Toys[2][0])

↓

OP = x

Date

Operators

↳ Arithmetic, Assignment, Comparison, Logical same as python,
slight extra +

(a) Assignment :

++x { increase or decrease by 1 } (unary Increment and decrement)
--x

(b) Comparison :

- == , console.log (1 == "1") O/P - True { Since J.s is weakly typed it does type coercion and both are string }
- === , console.log (1 === "1") ↓ (no type coercion occurs) O/P : false
- != , Conslole.log (1 != "1") O/P : false { Type coercion occurs }
- !== , Console.log (1 !== "1") O/P : True { No type coercion }

(c) Logical.

&& (and) (j == 1 && j == 2)
|| (or) (j == 1 || j == 2)
! ~ (Not) ! (j == k)

No, XOR operator, in js

Page No.
Date

- Escape character

e.g console.log("Hello\b World")

\b → move cursor one character position left / O/P: HellWorld
(o is omitted)

\f → to move cursor to next page

\r → to move cursor to beginning of line

\t → 4 space

\n → new line

\\ → for printing backslash (\)

\xxx → for ASCII character in octal form → \251 → ©

\xXX → " " " " Hexadecimal " → \xA9 → ©

\uXXXX → for Unicode character.

e.g - \u03A9

O/P: Ω

- Variable typing

- ~~typeof~~ n = "11"

typeof n , O/P → String (typeof tells variable data type)

- n = "123"

n += 1 → // Convert 'n' to number)

- n = 123

n += "" — // Convert 'n' to string)

explicitly

- n = "123"

n = parseInt(n)

n = parseFloat(n)

n = 123

n = n.toString()

- **Functions**

```
function products (a,b)
{ return a*b }
```

- **Global Variables**

Defined → Outside of any function +

defined within func^n, but without var keyword.

- also, variable declared within function without var or let, have global scope.

- **Local Variable**

Defined → Variables declared Inside a function or within a particular block are considered local variables.

Eg
```
function test()
{ a = 123;          // global scope
  var a = 123;      // local scope
  if (a==123) var c = 789 }   // local scope
```

- **Difference in var and let**

- Variables declared with var have function-level scope
- They can be re-declared and updated.
- Variables declared with let have block level scope.
- They can't be re-declared within same scope.

- Using let and const

var: Variables can be re-declared and updated.

let: Variables can't be redeclared but can be updated.

const: Variables can't be re-declared or updated.

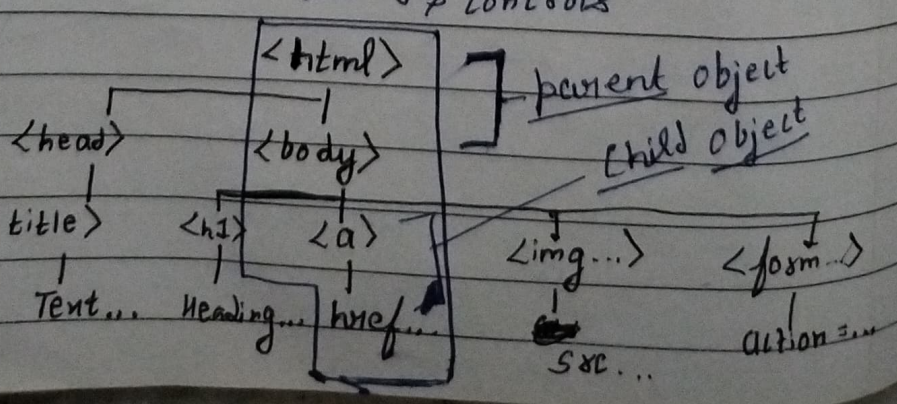
- Both var and let can be declared without being initialized but const must be initialized ~~without~~ during declaration.

- When var is declared without being initialized it automatically initialized with undefined, but let variable goes into temporal dead zone until they're explicitly initialized.

The Document Object Model (DOM)

↳ • It break down the HTML Document into discrete objects

↳ each with its own properties, Methods, and JS controls



eg

```
<body>
  <a id="mylink" href="https://mysite.com"> click me </a>
  <script>
    url = document.links.mylink.href
  </script>
</body>
```

It refers to <html> and <body> tags

refers to <a> tag

refers to href attribute

- So, how code follows DOM tree
  - ↳ document → links → mylink → href

also, there is short form

url = mylink.href

Another use for $ Symbol

$ symbol has also meaning in js (other than jQuery)

$ → document.getElementById

e.g → document.getElementById('mylink').href

↓ replace by this

$('mylink').href

- Using the DOM

links object is an array of URLs

- url = document.links[0].href

(for first Url)

- numlinks = document.links.length

↳ for giving number of links in entire HTML document.

- for (j=0; j< document.links.length; ++j)
{ document.write (document.links[j].href + "
") }

↳ for accessing all links in document.

- Expression:

↳ combination of ~~letter~~, values, variable, operator and function etc

↳ document.write ("a: " + (42>3) + '<br')

O/P ⇒ a: true

- Literals and Variables:

- Literal is simplest form of an expression, which evaluate to itself

example: 42 (Numeric Literal)

"Hi" (String literal)

true (constant literal)

Page No.
Date

- Variables are also expression, which evaluates to the value assigned to it.

```
myname = "Ram"  (String Variable)
myage = "20"    (Numeric Variable)
```

- Operators and their precedence

↳ unary, binary, ternary
(++a, -a) (a+b, a-b) (?x:y,0)

Precedence, (High to Low) (& Associativity)

() [] .	(parentheses, call, and member)	
++ --	(Increment, decrement)	(None)
+ - ~ !	(Unary, bitwise, and logical)	(Right)
* / %		(Left)
+ -		(Left)
<, >, <=, >=	(comparison)	(Left)
==, !=, ===		
& ^ |	(Bitwise)	
&&	(logical)	
? :	(ternary)	(Right)
=, +=, -=, *=, |=	(Assignment)	(Right)
,	(separator)	(Left)

- Operator's Associativity: Direction of processing
  ~~sometim~~ Mostly it's, → left to right
  but in some → right to left case

e.g. level = score = time = 0
right to left →

- **Logical operator**

The || (or operator) can cause unintentional problem

because, the second operand will not be evaluated if first is True.

e.g `if (finished == 1 || getnext() == 1){done = 1;}`

↑ So it will not evaluated until, finished == 1 is true.

for avoiding this we can use =>

`if (finished == 1 OR getnext() == 1 ){done = 1;}`

- **The with Statement**

It reduces many references to an object to just one reference.

e.g
```
<script>
String = "Hello, How are you?"
with (String)
{ document.write( "String has" + length + " characters (incl. spaces)");
  document.write( "In upper case:" + toUpperCase())
</script>
```

O/P => String has 18 characters {including spaces}
In upper case : HELLO, HOW ARE YOU.

Page No.
Date

~~Here~~, Since, string is not directly referenced by document.write, but still it referenced it using with (String) statement.

- The ternary operator (?)
  - ↳ quick way of writing if. else....

```
document.write (
  a<=5 ? "a is less than or equal to 5" : "a is greater than 5")
```

If It's True then this will, otherwise this will execute execute,

↳ If (a<=5) { document.write ("a is less ... 5")
else { document.write (" a is greater than 5")

prefix Increment → first Increment, then use updated Value. → (++b)
postfix Increment → first use value, then increment. ↳ (b++)

eg let x = 5
let y = ++x { first x = 6, then y = 6 }

let z = ++5 { first z = 6, then y = 7 }

Date

do-while loop { If you wanted your loop to work at least once }

count = 1 → Initial

do { document.write (count++ + "
") } → run once

while (++count <= 7)

↳ than loop start in which Increment + Breaking condition

- **for loops**

for (initialization ; condition ; Increment / decrement)

{ what to execute }

double loop

for (i=1, j=1 ; i<10 ; i++, --j)

Break / Continue

- **Explicit Casting**

Integer	parseInt()
Boolean	Boolean()
float	parseFloat()
String	String()
array	split()

Page No.
Date

let c = "Car"

- displayItems ("Bananas", 32.3, c)

function displayItems()

using arguments array

{ for (j=0; j< displayItems.arguments.length; ++j)
document.write (displayItems.arguments[j] + "
") }

Introduction to jquery

- **JQuery:**
  - It provides very high level cross browser compatibility
  - Also easy and quick way to access HTML and DOM manipulation, special functions to directly interact with css
  - Powerful animations + asynchronous communication with web-server
  - It's also a base for wide range of plugins and other utilities

- **Including JQuery:**

  Way 1: Download the version you need, upload it to your web-server, and reference from a <script> tag in your HTML files

  Way 2: Take advantage of a free C.D.N and simply link to the version you require.

  { It's an MIT license project, so there is no restrictions on what one can do with it }

- 3 flavours of jquery: 1.x, 2.x and 3.x { 3.7.1 → Stable } latest → 4.x (beta form)

- Diff in compressed or editable:
  - minified version (• Compressed in size to minimize bandwidth and decrease loading time)
    - It also remove comments
  - uncompressed version → you can edit yourself to it

- We also customized "own jquery" using jQuery Builder

- **jQuery Syntax**

  '$' Symbol acts as jQuery factory method

  → the main means of accessing into the framework

  or

  one can directly use 'jQuery()', function.

General Syntax → $('Selector'). period or method on selected element

example :

```
<body>
    The jQuery library uses either the <code> $() </code>
    or <code> jQuery </code> function names.
</body>
<script>
    $('code').css('border', '1px solid #aaa')
</script>
```

- **Avoiding library conflicts**

- Some-times other library try to use same '$' symbol. which can cause conflicts
- To fix this you can use → $.noConflict() which can tell jQuery to give up control of "$" symbol so other library can use it.

Also, to avoid this you can use 'jQuery function'

or, you can replace $ with object name,

jq = $.noConflict()

↳ now you can use jq instead of $.

Selectors

↳ element
↳ Id
↳ Class
↳ Combining Selector

css method

.css ('property': 'value';) ← { It dynamically alter any css property }

Eg $('p').css ('text-align':'center';)

← { It is also use to return a computed value }

like» Hello

↳ and if you wanted to retrieve color or anything of that element use css method without passing second argument.

var color = $('#abc').css ('color');

O/P => colors of #abc in rgb format.

- jQuery return value, which gets computed by jQuery at the moment, the method is called
  ↳ but it's not the original value!

  e.g: If originally color was set as
  color: blue, or
  color: #0000ff, or
  color: #00f
  but it will return
  rgb(0,0,255)

- also, we can use that computed value, again
  eg: var shade = $('p').css('color')
  O/P of shad → rgb(0,0,255)
  Now, Let second element with id #xyz, and if we want to assign color of #abc, we do this →

  ~~$('#xyz').css('color', shade);~~

  $('#xyz').css('color', shade);

- atlast, we also pass multiple css property via object literal

```
$('#abc').css({ 'color': 'blue',
                'font-size': '16px',
                'font-weight': 'bold' });
```

- Combining Selector:

$('blockquote, #abc, .xyz').css('color': 'blue');
or '#00f'

Event Handling

e.g
```
<body>
  <button id="btn"> Click me </button>
  <p id='result'> I am para </p>

  <script>

    $('#btn').click( function ()
    { $('#result').html ('You clicked button')
    })
  </script>
```

before

Click me

I am para

after

Click me

You clicked button

when using event, omit on
like onclick, ondblclick
→ click, dblclick

Date

- Always use → `$('document').ready(function(){`
  `// Your code here`
  `})`

  or Just

  `$(function(){`
  `—— code ——`
  `})`

It's best to put → • cdn link of jquery in <head>
• Inline Script to put at end of <body>
• and jquery code in document.ready
→ to get best result of js

- focus and Blur event

`$('input').focus(function(){ $(this).css('color','#fff0') })`

`$('input').blur( " " - - )`

- Click and dblclick

↳ also we add effect like → slideUp(), hide()

`$('.abc').click( function(){ $('this').slideUp() })`
`" .dblclick " " .hide()})`

{hide, + show} → toggle

- **Keypress event**

```
<script>
    $(document).keypress(function(event)
{
    key = String.fromCharCode(event.which)
    if (key >= 'a' && key <= 'z' ||
        key >= 'A' && key <= 'Z' ||
        key >= '0' && key <= '9')
    { $('#result').html('You pressed' + key)
      event.preventDefault() }
})
```

→ (convert the code provided by which to single letter.

↳ normalization of which tells which key is pressed

↳ It prevent normal action of key, and helps to print it

Not use document.write(),
because by the time key is pressed, document is fully loaded, so if we use that it over-write ~~ente~~ entire document

Mouse-event

- Mouseenter – mouseleave
- mouseover – mouseout } do same thing
- hover ——— ↗ to combine these

e.g $('#Id').mouseover(function(){ $(this).html("...") })
$ " .mouseout (" " " })

or

$('#Id').hover(function of mouseover },
function of mouseout })

or using chaining {period operator to combine}

$('#Id').mouseover(function(){ _____ }).mouseout
(function(){ })

- Submit event

$('#form').submit(function(){
alert('form submitted') })

- Select event

$('input[type='text']').select(function(){
alert("Text selected!"); });