— {Alt + Tap}

HTML

* Hyper Text Markup Language. HTML is the standa markup Language for webpage. 4 required tags in HTML is html, title, head and body.

* NOT a programming Language.

Refer
* mdn
* W3School

1) Document title inside the title tag.
2) The page title inside the head tag.

Tag

* bit of text. Act - point demarcation. putting Character in angle brackets. For Example, <h1> .... </h1>.

* Total Tags in HTML - 119.

Syntax:

<!DocType HTML> — Version type. (not a case sensi

16px - <html> — root element.

<head>

<Meta></Meta>

<link> Void element

<title></title>.

</head>.

<body> </body>.

</html>

Lab:
* 1200px - View Point.

Tag with content is called element

Command syntax: <!-- commands -->.

Void element: No close tag.

Ex: * img tag * opt group.
* link tag * area
* hr tag
* br tag

<html>
All other elements m be decendan of this element.

URL- Uniform Resource Locator.

BLOCK element

* Take whole width in view port.
* Always start a new line.

Inline element

* They will display in a line. occupy only needed Spaces.
* NOT start with new line.

Inline / Block element

* audio * video * img

Formatting elements

* Set the text style. For Example, * b tag * i tag * em tag * q tag * sub tag * sup tag.

Sematic element

Inline element.

* It clearly describe meaning to both browser and developer. For Example, * Strong * form * table * article * figure * main

Attributes - Modify Effect

* provide additional information about HTML element.
* Always in the open tag. Pieces of Markup Language.

Syntax: attributes name = "value".

Ex: * id = identify the element uniquely.

* class = associate with the a style sheet by specifying the class of element.
* title = Specifies extra information about an element.

1) <abbr> HTML </abbr>

Output: HTML

2) <abbr title="Hyper text markup Language"> HTML </abbr>

Output: HTML

Hyper text markup Language

Boolean - element attribute basically means true (or) false condition. true-1 false-0

Form / Input

1) open - dialog tag (use)

2) required *disabled *enabled *checked

3) controls *mute *loop - audio and video tag.

4) reversed.

Event Attribute: ability to let event trigger action browser.

*Java script event. Ex: *Onclick *onsubmit. *alert *On mouse in out

---

1) figure tag - intent அகி இருக்கும்.

Syntax:

```
<figure>
    <figcaption> Flower. </figcaption>
    <img src="image/flower.jpg">
</figure>.
```

Syntax:

```
<picture>
    <source media="min-width:300px"
        src="image/flower.jpg">
    <img src="image/flower.jpg" alt="loading">
</picture>
```

*Picture tag use பண்ணும் different view port-ல adopt ஆகிக்கும். like phone, Labtop.

target - specifies where to open the linked document

13/6/2023

Heading tag - size

* h1 - 32px / css - 2em
* h2 - 24px / css - 1.5em
* h3 - 18.72px / css - 1.17em
* h4 - 16px / css - 1em
* h5 - 13.28px / css - 0.83 em
* h6 - 10px / css - 0.75em

1) b - bold not important.	Strong - bold important.
2) del - delete in Browser.	s - no longer correct.
3) i - italic.	em - italic emphasized text.

14/6/2023

Meta tag

*inside the head tag. used to *character set. *Page discription. *keyword *author of document. *viewport setting. UTF - Unicode Transformation format

Attributes Type

* Global * Boolean * Event.

Global: used in all tags. Ex: *height *width *class *name *lang *id* *style.

class = . id = #

Element selector

```
<style>
    h1 { back-ground-color: Red }
</style>
```

specify - id tag"

class - More than one.

Ex: `<h1 title = "hai"> welcome </h1>`

output: welcome

hai

Tabel - element - Depends on row.

Syntax:

```
<table>
    <caption> Table </caption>
    <tr>
        <th> </th>  → box center-ல இருக்கும்.
        <td> </td>
    </tr>
</table>.
```

Attributes Table

* border
* Cellspacing = "0"
* Colspan - column merge
* rowspan - row merge.
* border_radius : 10px - ▢
* border_collapse

Form elements 19/6/2023

Syntax:

```
<form>
    <field set>
        <legend> </legend>
        <label For = "1" > male </label>
        <input type = "radio type" id = "1" >  ← Void element
    </field set>
</form>.
```

Value = Specifies the value of an <input> element. put actual content inside the form.

*Placeholder = Specifies a short hint that describes the expected value of an input field.

---

difference

Value = if you want to change it you have to delete first.

Placeholder = When you cli on the box and start typing it disapears.

15/6/2023

HTML - List

* Ordered list * unordered list * Description list.

Syntax:
```
<ol>
    <li> Tamil </li>
</ol>
```

Out put: 1. Tamil *default is • - disc. [ul tag]

Discription list

Ex: HTML - Term
Hyper text markup Language - data / defn.

Syntax:
```
<dl>
    <dt> HTML </dt>
    <dd> Hyper Text markup Language </dd>
</dl>
```

Details, summary Tag

Syntax:
```
<details> DS
    <summary> Data science </summary>
</details>
```

output: DS
Data science

Select Tag

syntax:
```
<select>
    <label for="pro"> Select your Dep </label>
    <select id="pro"> → void elements
    <opt group label="dep"> → void element
    <option value="maths"> maths </option>
    <option value="English"> English </option>.
</select>.
```

output: Select your Department [maths ▼]

- Science
- Maths
- English

audio tag

```
<audio controls>
    <source src="audio/1.mp3" type="audio/mp3">
    <source src="audio/1.mp4" type="audio/mp4">
    <source src="audio/1.ogg" type="audio/ogg">
</audio>.
```

<embed> - Audio, Video tag-ஐ use பண்ணலாம்.

<iframe> - oyer frame-ஐ எதுவேணும்னாலும் display பண்ணலாம். <iframe src="" />

```
<a href="URL" target="myframe/blank"> google </a>
```

output: google.

---

1) <button> click me </button> ⇒ [click me]

2) <button disabled> clickme </button> ⇒ [Click me]

---

progress tag - dynamic

```
<lable for="pro"> progress bar: </label>
<progress id="pro" min="0" max="100" value="50"> 50
</progress>.
```

output: progress bar: ▬

without value. it is moving.

Meter tag - Statics

Value போட்டாலும், போடாவிட்டாலும் Move ஆகாது.

Meter - used for such as disk space, memory usage

---

progress - display the progress of the specify task.

Radio input-type

Syntax:

```
<label for="1"> Gender: </label>.

<input id="1" type="radio" name="2"> male

<input id="1" type="radio" name="2"> female.
```

OUTPUT: Gender : male ○ Female ○

* Name - எல்லாத்துக்கும் ஒரே மாதிரி இருக்கனும். அப்போதான் ஒன்னை மட்டும் Select பண்ண முடியும்.
* without name - எல்லாத்தையும் Select பண்ணலாம்.

Form Attributes

1) action - progress after submit button.

2) target = value = -blank , -self , -parent.

use:

3) Method = Value = get -

post -

4) auto complete - Value - on , off

5) no validate → Boolean Attribute

input attribute

1) name -

2) type -

3) placeholder -

4) id -

3) * button -

button Create
பண்ணலாம்.

* Submit -

* Reset -

4) * range -

* radio -

* Checkbox -

* file -

5) * Image -

* color -

* URL -

* hidden -

* Search -

```
<div>
</div>
```

style:

```
border-bottom : 30px
                solid black
border-right : 20px
               solid Traspaen
border-left: 20px solid
             Transparent;
```

range input

Syntax:

```
<label for="i"> Review </label>
<input range>
<input type="range" id="i">
```

OUTPUT: Review

Syntax: Datalist

```
<label for="1"></label>
<input id="1" list="2">
<datalist id="2">
<option value="Rose">
<option Value="Jasmin">
</datalist>.
```

} pre defined (No close tag).

4)

Input Type - 22 type

1) * text - single-line text field. default width of the text field is 20.

* e-mail - defines a field for an e-mail address.

5)

* password - defines a Password field. (Character are masked).
* number - field for entering a number.
* tel - field for entering a telephone number.

2) * date - define a date pickers. The resulting value include the year.

* time - define a control for entering a time. (no time zone).
* week - defines a week and year control (no time zone).
* month - defines a month and year control The format is "yyyy-MM"
* datetime-local - defines a date picker. The resulting value include the year, month, day, time

1) git clone "copy" → add files → git add index.html → git commit "add new" → git push.

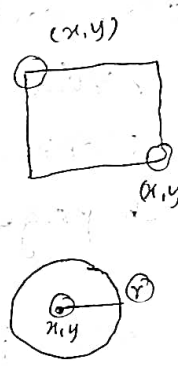
Form style tag use பண்ணும்போது → <style> tag

* canvas { float: right }
* .class { margin-right: 20px }
* attribut name { display: inline-block; padding-top: 20px }

Map tag

Syntax:

```
<img  src=" "  alt=" "  usemap="#my map">
<map  name="my map"
<area  shape="rect (or) crl"
       coords="x, y, x, y" (or) "x, y, r"
                rect          crl
       href=" "  target "self" >
</map>
```



Meta tag

* data about data. This is a void element.

website - data

Syntax:

```
<meta  Charset="UTF-8" >
<meta  name="viewport"  → phone, Lap adopt.
       content="width=device-width,
       intial-scale="1.0" >
refresh
↑ <meta  http-equiv="refresh"
         content="5"
         Url="https/www.fabevy.com" >
<meta  keywords "name="keywords"
       content=" html, H, It"
```

UTF-8 = Unicode Transformation Form
↓
encoding and decoding.

UTF-8 (or) ASCII

hyper text Transfer protocal

Icon format

Syntax:

```
<!DocType html>
<html>
  <head>
    <title> </title>
```

Favicon ← `<link rel="icon" href="image/fabevy.jpg">`

`<link rel="stylesheet" href="font awesome">`

`<style> .icon { font-size : 40px } . </style>`

```
  </head>
  <body>
    <i class="fa fa-youtube icon"></i>
  </body>
</html>
```

must

Content management System : - git (provide) Desktop App

git Hub - Web appli

↓

Version control

1) □ - create repository.
2) clone ← 1) Time
3) add files
4) git add
5) git commit
6) git push:

Practice → git practice → Right click → Git bash →

git clone "copy ctrl+c paste" ctrl+v Enter → git status (No use)

↓ New file open ctrl+n.

Now git practice → Right Click → New → Newfile → text document →

5) value -

6) min -

7) max -

8) max-length -

9) auto complete -

10) size -

11) novalidate -

*12) required -

13) disabled -

14) enabled -

15) encode -

---

link and script Tag

↓ head (or) body - இரண்டு பயன்படுத்தலாம்.

best body - க்கு முன்

Syntax:

<link rel=" " href=" "> → void element.

rel = "style sheet"

href = " folder name/style.css".

↓

W3School → Font awesome intro → Basic icons

copy பண்ணிக்கணும் ~~body~~ head - ல paste பண்ணனும் → best use (4.7.0 versio

---

Ex: <body>

<i class = "fa fa-youtube icon">

</body>

<head> → class

.icon { font-size : 40 px }

</head>.

CSS- Cascode Style Sheet - Version-3

Used to define styles for your web pages, includi the design, layout and variations in display for different devices and screen sizes.

3 type:

1) Inline style - styles that applied to a specific element within the body section of webpage.

* The style will be applied to that individu element only rather than to entire page (internal style). Quick and specific style.

The first priority

Ex: style attribute - Global attribute.

2) Internal style: used for multiple elements within the same HTML document. It is called embedded css.

* style tag inside the head tag.

* The rule only apply to the page.

Second priority.

3) External style:

Syntax: <link rel = "stylesheet" href = "css/style.css"

Separate css file that can be accessed by creating a link within the head section of the web page.

change the entire website by changing just one files.

CSS- background properties:

1) background-color:

2) background-image: url(../image/image1.JPS).

3) background-size; cover
contain (default).

4) background-position : top ; bottom ; center.

5) background-repeat: no-repeat ; repeat; repeat-x;
repeet-y;

6) background-attachment: fixed ; scroll (default).

7) background: linear-gradient(red, green),
url(image1/flower.jps); (short hand property).

8) background-blend-mode: multiply;

---

media-css-save- Media Quary;

@ Media

Syntax: @ media (minwidth : 320px){
.container{ width:100%. }
}

mobile:

ipad: @ media (min-width :660px){
.container{ width :100%. }
.row{ display :flex ;
flex-wrap : wrap;}.
.card { width : 50%. }
padding: 20px;}
}

Lap: @ media (min-width : 960px){
.container{ width : 60%.
margin : auto }.
}

Lap : 1200px;
Mobile: min 320p
iPad : min 660 px
Mobile: min 960

<meta name="author" content="APTA"

<meta name="description" content=" "

↓

content about website

Entities like symbol, Letters, emoji

It is a code

---

link tag

you can link the Html page, css, and any weblink.

* void element.
* connection between two web page.

a tag

used to insert a another source link.

* Not a void element
* you touch the link, it open in New tap.

---

meta purpose:

Refer to the text, that is displayed on search engine result pages.

---

without Doctype:

It is work. But Validation error.

HTML

```
<div>
    <h1> Head </h1>
    <p> para-1 </p>
    <p> para-2 </p>
    <span>
        <h3> head-2 </h3>
        <p> para-3 </p>
    </span>
    <p> para-4 </p>
</div>
    <p> para-5 </p>
    <p> para-6 </p>
    <p> para-7 </p>
```

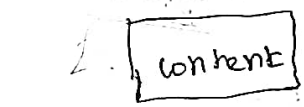
CSS

```
* {
    margin : 0;
    padding : 0;
    box-sizing : border-
                 size;
}
```

Box Model: Padding - inner space

* Padding : 10px → 4 side-ல
* padding : 10px 5px 20 px
  Top | Left, Right | bottom
* padding : 10px 5px 12px 25px.
  Top | Right | Bottom | Left

} Shorthand property.



* padding-right : 10px.

Margin - Outer space

5+5 = 10px

* margin : 5px
* margin : Auto → Block element-க்கு மட்டும் use பண்ணலாம். inline-ல use பண்ண முடியாது.
  ↓ width கொடுத்தா தான் Changes பண்ண முடியும்.

Center-ல கொண்டு வரலாம்.

1) Descendant selector: - symbol - space.

Select all paragraph. represented by single space.

Ex: div p { color : pink ; back-ground : yellow ; } → Take all p tag.

2) child selector: - symbol - >

used to match all the elements which are children of a specified element. It gives relation between two elements. Child மட்டும் எடுக்கும். grandchild-ஐ எடுக்காது.

Ex: div > p { color : yellow } → Take only child p. not grand child p.

3) Adjacent Sibling:

used to select an element that is directly after another specific element.

பக்கத்தில் இருந்தா எடுக்கும். ஒரு tag-ஐ தாண்டி எடுக்காது.

Ex: div + p → div அருகில் உள்ள p மட்டும் எடுக்கும். ஒரு வேளை இடையில் hi இருந்தா அதை எதையும் consider பண்ணாது.

4) General Sibling:

Selects all elements that are next siblings of a specified element.

Ex: div ~ p. → div அருகில் எந்த இடத்தில் p இருந்தாலும் எடுத்துக்கும்.

Simple:
* id
* class
* element

priority:
1) * D Child selector
   * Descendat
2) * General
   * Adjacent

* Combinator தான் 1st priority.
next தான் Simple-க்கு.

css - command : /* commands */ shortcut : ctrl + /
ctrl + q.

save css : style.css syntax : selector { property : value; }

CSS - selector :

first part of css Rule. It is a pattern of elements and other term that tell the browser which HTML elements should be selected to have the css property value inside the rule applied to them.

Type :

* simple selector, group selector. * combinator
* pseudo - class * pseudo - elements.
* universal selector (*)

Simple selector :

select elements based on name, #id, #clas

Ex: * Element name [h, p, img, div] - 3 priority.

1st priority * id [#] → * Class [.] → specificity 2 - high priority.

group selector : To change a default settings.

universal : * used to select all the elements in HTML.

universal selector. * Asterisk (*) symbol denote [margin, padding : 0].

combinator : select elements based on a specific relationship between them. (the elements)

4 Type :

* decendant selector - symbol - spa
* Child selector - symbol - >
* Adjacent Siblings - symbol - +
* General Sibling - symbol - ~

```
<div>
    <p>...</p>
    <h1>...</h1>
    <p>...</p>
</div>
<div>
    <p>...</p>
    <a>...</a>
</div>
```

: first-child
: last-child
: nth-child.
: disabled
: checked.

* first_child (2) → first of 2nd child.
* last-child (2) → last of 2nd child.
* nth_child (odd) → odd no. 2nd child.

CSS - display:

display : block; inline; inline-block; flex; inline-flex. none; grid

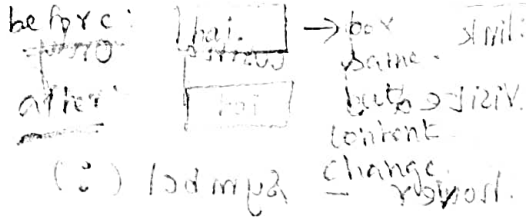
difference

box-sizing : content box

* content size same while border box size grows as padding and border grow.

box-sizing : border-box

* border-box same while content size decreases padding and border grow.



display:none

visibility:hidden

blur - ... - Letter double ... offset.

set - 0 0 → 4 side

different between:

Box-shadow	text-shadow
without shadow color, default color is Black.	without shadow color, It takes text-colour.

different between:

Margin:

Collapse between two elements. and it takes highest margin value of the element.

Padding:

adding two elements (between). add padding value of the element.

—html:

```
<h2> head </h2>
```

output:

head

CSS

```
h2 { text-decoration-line: underline overline; line-through;
     text-decoration-style: double
     text-decoration-color: Red }.
```

style. depend on line

Pseudo-class(:) — order

* :link
* :visited
} using only a tag.

* active - fraction of time - ... change

* :hover - symbol (:) → ... touch ...

syntax: 1) a : link { color : red; } → unvisited link.

2) a : visited color : Green → ... visit ... link.

hover: Take place when a user hovers over an elemer and element responds with transition effect.

Text - property:

1) color

2) text-align: center; Left; right; Justify.

3) text-decoration line/style: solid → always none.

4) Text - Indent: 10px; 2%

5) line-height: 1; 2;

6) word - spacing:

7) Letter- Spacing:

8) text-transform; Capitalize; Upper case; Lower case.

---

font property.

500 வரை no-changes

1) font-Size: 20px/large/small larger

2) font - weight: 100-900/ bold/bold (thickness, no unit (px...))

2) font-family: default: Poppins, times-newRomen

4) font-style: italic/normal. (default)

larger- depend on parent. (parent பொறுத்து மாறும்)

Colors:

alpha-Value transparent-ஆ காட்டும் (வெளிர் நிறம்)

* colour name
* rgb = red green blue.

Syntax: Color: rgb(250, 200, 178) (or) hsl (degree)

hsl (150deg, 50%, 10%)

hue, saturation, light

* rgba → alpha Value.

Syntax: Color: rgba (250, 200, 178, 0.5)

(0, 250)

alpha range: 0-1 ⇒ 0 - Black, 1 - original

status

Hexa - Code: → status 0-9, A-F (or) (a-f)

Syntax: color: #0000 - black / #ffff - white.

*color picker (search).

* box-shadow: 5px 5px — (x, y)

---

* box-shadow: 5px 10px 15px red inset

x obsect, y obsect, blur, color, உள்ளே தெரியும்

4 பக்கமும் shadow தெரிய-வேண்டுமானால் x, y-ல Zero கொடுத்துவிட்டு, blur-ல value-ஐ அதிகமாக வைக்கவும்.

* position : sticky;

Use { top : Left ; Z-index : 100px ;

bottom, right → not use

* opacity → level (0-1) (Transparent).
↓ dark ↓ original

CSS - filter:

(white coating) (Black coating) (degree)
* Constrast (%) * brightness * hue-rotate → color change

* Blur - (px) * Gray-scale (%)

* Opacity (0.5) * invert - Like a scan image (-ve film)

* saturate (%) - Gray color layer.

* Sepia (%) - Brown color shadow.

* Drop shadow - B

Syntax:

Ex:
```
img {
    filter : blur (10px);
    filter : brightness (10%);
    filter : drop-shadow ( 0px  5px  5px  red ,
                           0px  5px  5px  yellow).
```

Transition: Smooth effect:

1) delay:
2) duration :
3) property:
4) Timing function :

Shorthand property: transition

linear → ease-in , ease-out , ease-in-out , cubic-bazi
(0,0, 0.2, 1)

Syntax: .row (parent property):

width = 100%;

display: flex;

flex-direction: row;

Justify-content: space-evenly.

flex-wrap: no wrap / wrap.

align-item: → height

align-content: → wrap

Syntax: child (property)

align: self

stretch.

staring.

icon

<i>

display: inline-flex

align-content:
center;

Position - property:

* static (default)
* relative
* absolute
* fixed
* sticky

Syntax: 1) position: relative;
top: 20px;
left: 20px;

2) position: absolute;
top (or) bottom: 30%; → viewport (depend மாறும்)
Left (or) right: 40%;

depend change:
parent க்கு position: relative கொடுத்தா
அது depend ஆகி விடும்.

3) position: fixed;
same as absolute. but.
depend change ஆகாது (always). (view port) depend only,

flex - property:

```
<section id=s>
    <div class="container">
        <div class="s-row">
            <div class="s-col">
                <div class="s-card">
                    <div>
                        <img>
                        "
                        "
                        "
                </div>
            </div>
        </div>
    </div>
</section>
```

---

1) section: * background.

2) container: * width : 90%;
* margin : 0 auto;

3) .row (parent) : * display : flex; * flex-direction : row
column
row-reversed;
column-reversed

* justify-content: flex-start;
flex-end;
flex-center;
space-Between;
space-around;
space-evenly.

* flex-wrap : nowrap (default)
: wrap (change)

4) .col : width : 25%

:: Marker (pseudo-element)

Syntax:

```
<ol>
  <li> one </li>
  <li> two </li>
</ol>
```

```
li::marker{
  color: Red
}
```

Output:

- one
- two

```
li { list-style-type : none;
```

Output:

one
two

```
<ol>
  <li><i icon></li>
  <li><i icon></li>
</ol>
```

```
li i {
  color : Brown
}
```

Output:

- ✓ one
- ✓ two

---

icon-finder-16px;

```
li {list-style-image: url(  )}
```

---

table { border-s Table:

```
Table{
  border-collapse : collapse }
```

Animation:

@ keyframes:

property: Shorthand = Animation: APTA 2s linear infinite;

* Atiimation - name :
* animation - duration :
* animation - timing - function : Linear
* animation - iteration - count :

Next: @keyframes APTA {

0% { color: Red ; translate : scale(0.5) ; }

20% { color : blue }

---

header → section:

logo Navigation link Get Start.

Syntax:

```
<body>
    <header>
        <div class="container">
            ① —— <div class="logo">
                    <img> </div>
            ② —— <nav>
                    <ul> <li> <a href="#banner">
                    </a> </li>
                    <li> <a> </a> </li>
                </nav>
            ③ —— <div> class="Get">
                    <a href="#"> Get Start </a>
                </div>
        </div>
    </header>
```

Syntax: transition: all;

~~transition-duration: 0.4s;~~

transition: all is linear is

Transform: → 2D + 3D

X, Y (move) (%, px)

* transform: translate( ) → translate X (or) translate (y),
: scale (1.5) (using small → big). → 0 இருந்தால் hidden
: skew
: rotate (deg)

overflow-hidden.

Pseudo-element:

symbol: ::

* before * after * Marker

Syntax: <button> Click </button>

button::after { }.

For Example: <h1> Heading </h1>.

CSS: h1 { position: relative }

h1::after { content: ""; → url

relative இருந்தால் 100%. ← { height: 2px; width: 100%.

(X) ← display: block;

border:
background-color:
Position: absolute;

for, while, do → loop.

function → function.

Data Types: Primitive and Reference.

* Null * undefined * Number * String * object
* NaN * BigInt * Boolean.

arithmetic operation

x = 10 + 10

output: 20

concordination.

y = "10" + 10

output: 1010

What Type:

var a = 10 → console.log (typeof a) ⇒ number.

var b = "APTA" → console.log (type of b) ⇒ string.

array – collection of items. → [ ] → object

category:

var c = [10, 20, 30, "APTA"] → array.

console.log (typeof c) → array object

object:

var d = { name : "hema", value → object

id : 10 } → key

console.log (typeof d) → object.

Big-Int

var x = 12345678910111212 (n)

NaN ⇒ a = "hai" * 10 ; output: not a number.

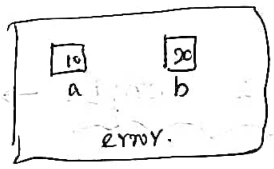
console.log("x=" +x); ⇒ x=30

Let → Block Scope. { } — 2015 use

Var → not Block Scope. → old Browser use since 1995.
Global or function

const. → Stable.

```
const a = 10;
const b = 20;
a = a+b;
```



const a; → initializing error.

* Names can contain letters, digits, underscores, and dollar signs.
* Names must begin with a letter. also start with $ and underscore.
* Names are case sensitive (a, A - different variable).
* Reserved words (Java keywords) cannot be used as names.

Do	Don't Do
* Let 1hema X	* let hema1 ✓
* Let *hema x	* let hema* ✓
* let <he	* let $hema ✓
* let +APTA x	* let _hema ✓

Reserved keywords:

Var, let, const → variable.
if, else, switch → condition.

text-align:

vertical-align: middle ; Top ; bottom.

color:

tr: nth child (1) {
background color }

(odd) (even)

Case sensible

Java Script: → user interaction.

↓ (1995)

develop: Brendan Eich

version: ES13

<script src="main.js">

</script>

</body>

line by line — Execute

reassigned

Variables:

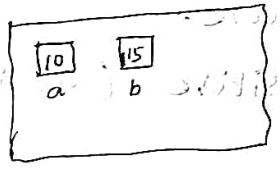
Variable name = value;

var a = 10;

variable declaration..

nothing → for loop.

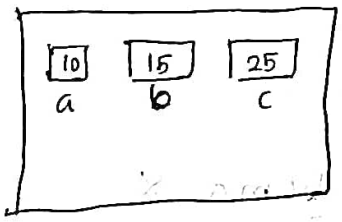
* Var
* Let
* ~~Var~~ Const



used as a container store a value.

Ex:

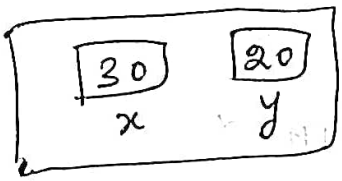
```
var a = 10;
var b = 15;
var c = ;
    c = a + b;
```



Ex:

```
var x = 10;
var y = 20;
var x = x+y;
console.log(x);
document.write(x);
```

} → output 2 lines.



Output

10
20
30

document space:

1) document.write("welcome" + "1");
document.write("welcome" + "2");
document.write("welcome" + "3");
document.write("welcome" + "4");

for Ex:

Output: welcome1 welcome2 welcome3 welcome4

2) document.write("1" + " "); → space.
document.write("2" + " ");

Output: 1 2

3) document.write("1" + "
");
document.write("2" + "
");

[+ nbsp;]

Output:
1
2

4) var a=16;
if (a>b) { → Block scope.
document("It is greater than 10");
}

ഒരു condition കൊടുക്കുന്നു,

else {
document.write("less than 10");
}

else → condition ഉണ്ടാകില്ല

a=10, b=5, c=3.

AND (a+b) && (b+c) && (c+a) → greaterthan / Lessthan } - true (or) false

output: 13 → Last 2 digit number மட்டும் வரும்.

but: OR (a+b) || (b+c) || (c+a)

output: 15 → first 2 digit no-மட்டும் வரும்.

!= (String வரும்போது)	!== (String வரும்போது)
a=10, b=10, c="10", d=5, f="5", g="10"	a!==b = false.
a!=b ⇒ False.	a!==d ⇒ True.
a!=d ⇒ True.	a!==c ⇒ True.
a!=c ⇒ false.	c!==f ⇒ True.
a!=f ⇒ True.	c!==g ⇒ false.

a = "hashini"; b = "20";

a>b ⇒ True.

a<b ⇒ false

[String - 2 digit number-ஐ விடுத்தாலும் அது letter's ஆக வரும்]

a = "hashini"; b = "2000021"

a>b ⇒ True.

a<b ⇒ false

Condition:

[if, else, else if, nested if] [switch]

If Syntax: if (condition) { }.

Input: var username = "hema";

Output:

```
if (username == "hema")
{
console.log("welcome");
document.write("wel");
```

Undefined:

var y; → console.log(typeof y) → Undefined.

Null:

var x = [ ] → console.log(typeof x) → null.

variable

Yes.

var a, b, c = 10, 20, 30;

Arithmetic Operators:

+	-	*	/	%	**	=
			quotient	remainder	power (exponential)	assign

, comma → separate

= → assigning operator.

Comparision - Operator:

== → equal (true (or) false) [value - மட்டும் தான் check]

a=5, b="5" → true

=== → (DataType → எல்லாம் check பண்ணும்] → a=5, b="5" false

!= → not equal to (value) → a=5, b="5", a != b → false

!== → Data Type a=5, b="5" a !== b → True

number & String → true.

Logical Operator:

AND && → All condition should be true

OR || → either one condition is true.

NOT !

another method:

~~if~~

if (a < a < 100)

Marriage Eligibility

male ⇒ 21 → elegible

female ⇒ 18 → Eligible.

~~var a=21, b=18;~~

```
var a = prompt ("Enter Gender");
var b = prompt ("Enter your age");

if (a == "male") {
        if (b >= 21) {
            document.write ("He is elegible
                for marriage");
        }
        else {
            document.write ("He is not
                eligible for
                marrige");
        }
}

else if (a == "female")
    { if (b >= 18) {
```

write a program to find given number is positive (or) Negative:

```
Var a=10;
if (a>0) {
        document.write ("Number is positive");
        }.
else if (a<0)
else {
        document.write ("Number is Negative");
        }

    else {
        document.write ("given number is zero");
        }.
```

---

write a program to find given number is two digit number?

```
var='23
if (a>=9){
        document.write ("number is 2digit
if ((a>9) && (a<100)
        {
        document.write ("a digit number");
        }.

    else {
        document.write ("not a 2digit number");
        }.
```

Syntax: If / else

```
if (condition) {
        }
else { }.
```

Syntax: else if

```
if (condition) { }
else if (condition) { }
    else { }
```

---

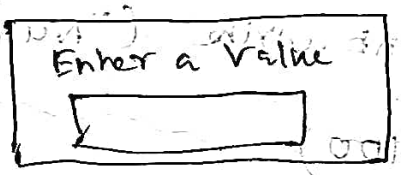
else if → Multiple time use

```
var a=10;
if (a>10) { document.write ("Greater than 10");
        }
else if (a<10)
        { document.write ("Less than 10");
        }
else {
    document.write ("equal to 10");
    }
```

Program: 1 build in function: → Put anything. It takes String.

1) var a = prompt ("Enter a Value")

Output page open:



alert ("Hello");

(not a decimal). String → Change ... (Integer ... Change ...).

2) var a = parseInt (prompt ("Enter a Value"));

Float Change ... (decimal ...)

3) var b = parseFloat (prompt ("Enter b Value");

program:

```
var a = prompt ("Enter 1st Value");
var operator = prompt ("Enter operator");
var b = prompt ("Enter 2nd Value");
var c;

switch (operator) {

    case "+" : c = a+b;
               console.log(c);
               break;

    case "-" : c = a-b;
               console.log(c);
               break;
}
```

Loops: Iteration

```
var a=1 ⟶ initialization;
while (a<=10) { ⟶ condition checking.
    console.log(a);
    } a++; ⟶ update.
}
```

Output:

1 2 3 4 5 6 7 8 9 10

Reversed

```
var a=10;
while (a >=1)
    {
    console.log (a);
    a--;
    }
```

Output:

10 9 8 7 6 5 4 3 2 1

```
else if ((c<a) && (c<b)){
        if (a<b) {
            console.log(c,a,b)
        }
        else (b<a) {
            console.log(c,b,a)
        }
    }.
```

Largest Number:

```
if((a<b) && (a<c)) {
        console.log(a + "is largest Number");
    }
else if ((b<a) && (b<c)) {
        console.log(b + "large");
    }.
else if ((c<a) && (c<b)){
        console.log(c + "large");
    }.
```

Switch Case:

```
switch () {
    case 1: ——
        break;
    case 2: ——
        break;
    case 3: ——
        break;
    case 4: ——
        break;
    default : ——
        break; }
```

```
{document.write ("she is eligible for
                  marriage");
}.
else {
    document.write ("she is not eligible");
}
}.
else {
  document.write ("Enter only Male (or) Female");
}.
```

Asscending order of 3 Numbers:

a,b,c	a big	b big	c big	decending. greather than symbol
	a,b,c a,c,b	b,a,c b,c,a	c,a,b c,b,a	

program:

```
if((a<b)&&(a<c)){
    if (b<c){
        console.log (a,b,c);
    }
    else {console.log (a,c,b);
    }
}
else if ((b<a)&&(b<c)) {
    if (a<c){
        console.log (c,a,b)
    }.
    else {
        console.log (c,b,a)
    }
}.
```

```
for (let i=1; i<=5; i++){
    document.write("*");
    document.write("<br>");
}
```

Output:

*
*
*
*
*

Inner Loop: String.fromCharCod(i+65);

```
for (let i=1, i<=5; i++){
    for (let j=1; j<=i; j+1)
    {
        document.write(i+" ");
    }
    document.write("<br>");
}
```

Output:

1
1 2
3 3 3
4 4 4 4

```
var n =prompt("Enter");
for (let i=1; i<=n; i++){
    for (let j=n; j>i; j--){
        document.write("  ");
    }
    for (let k=1; k<=i; k++){
        document.write("* ");
    }
    document.write("<br>");
}
```

Output:

. . . . *
. . . * *
. . * * *
. * * * *
* * * * *

Reversed Number:

```
var a = [10, 15, 20, 25, 30];
a.reverse();
```

Output: 30, 25, 20, 15, 10

Full → print ఎలా: upto-5

```
var j=1
for(i=1 ; i<=5; i++){
    j=j*i;
    document.write(j+"<br>");
}
```

Output:
1
2
6
24
120

```
var j=1.
var n = prompt ("To find the factorial number");
for (i=1 ; i<=n ; i++) {
    j=j*i; }
document.write (j);
}
```

Fibonanci series:

```
var n=prompt ("which times you want");
var a=0, b=1;
document.write(a+" "+b);
for ( i=1; i<=n ; i++ ) {
    c=a+b;
    a=b; b=c;
    document.write (" "+c);
}
}
```

Out put:

0 1 1 2 3 5

Pattern:

```
for (let i=1 ; i<=5 ; i++) {
    document.write (i);
    document.write ("<br>");
}
```

Multiplication table

```
var a = 2;
var b = 1; → initializing.
  var = c;
while (b <= 10) {
        c = b * a;
console.log (b + "x" + a + "=" + c);
        a++
}
```

b ≤ n

```
var n =
  prompt ("limit");
var a = prompt
  ("table")
```

For loop:

```
var a = 2; var c;
for (let b=1; b<=10; b++) {
        c = b * a;
console.log (c);
}
```

Even Number: from 1 – 10

```
for (let a=1 ; a<=10 ; a++){
  if (a%2 == 0) {
    document.write (a + "<br>");
  }
}
```

→ `if (a%2 == 1) { odd } else { even }`

0

Factorial Number Find = 5!

n=5

```
var j=1
for (let i=1 ; i<=5 ; i++){
        j = j * i;
}
document.write (j);
}
```

1 * 1 = 1
1 * 2 = 2
2 * 3 = 6
6 * 4 = 24
24 * 5 = 120

output:

120

Choose a particular Number in array: → 40 or 16

array's index / position.

Ex: a 2 arra 40:

Syntax: a[i]

a[0]=10 a[1]=20 a[2]=40

a[3]=30 a[4]="APTA"

Create a Array:

1) var a = [10, 20, 30, 40, "APTA"];

2) var a = [];

var b = prompt ("Enter the array Value");

for (let i=0 ; i < b ; i++) {

var a[i] = prompt ("Enter " + " i+1 " + " Value")

}

Console.log (a);

Delete Array: (or) empty.

var arr = [1,2,3,4,5,7,8,10]

console.log (a.length);

output: 8

1) Input: arr.length=0;

console.log (a.length);

output: [] → empty.

2) var arr=[];

console.log (a.length)

output: [] → empty.

Find odd Number:

output: 11 75 -9

var arr = [11, 10, 70, 75, 59, 2, 9];

count = 0;

for (let i=0 ; i < arr.length ; i++) {

if (arr[i] % 2 != 0) {

document.write (arr[i] + "
");

```
else {      a = "not even number";
    console.log(a);
    return a;
}

var b = findeven(5);
    document.write(b);
```

output:

Not a even number.

var	let	const
* function Scope	* Block Scope	* Block scope.
* Updated and declared into the Scope.	* Updated and not re-declared.	* cannot update and declared.
* declared without initializing	* cannot declared without initializing	* cannot declare without initializing
* accessed without initializing as default value is undefined.	* not accessed without initializing reference error	* not accessed without initializing initializing error.

Array → collection of elements

CRUD Operation: → create Read update delete

* Data Type: Number and String.

var a = [10, 20, 40, 30, "APTA"].

1) No. of elements:

a.length = 5

* It store Number, String.

* index/position always start with zero

Function:

Syntax:

```
function  functionname. (parameter1, parameter2
{
Let  answer = parameter1 + parameter2
     return answer;
}
```

Likewise:

```
function  add (x,y){
     Let c = x+y;
     console.log(c);
     return c;
}
```

Call: (or) invoke:

```
var a = add (2,9){
console.log(a);
}
```

Output: 11

function name - call pandren.

Function - ...
function-...(argumen
& pass ...
Call (or) invoke
...
Call Back funct

Scope:

*Global scope *local scope *Block scope *function scop

Built-in function:

*parse int *parse float *ISNAN *to Fixed

Find Even Number:

```
function  find even (parameter1) {
     if (parameter1 % 2 == 0) {
          a = "even number"
          console.log(a);
```

Compain:

```
var a = [1,2,3,4,5];
var b = [6,7,8,9];
for (let i=0 ; i < b.length ; i++){
    a.push (b[i]);
}
document.write (a);
```

Ascending Order of Array:

```
var a = [25, 17, -48, 12, -39, 79];
var result = [];
var temp = 0;
for (i=0 ; i < a.length ; i++) {
    for (let j=i+1; j < a.length ; j++){
        if (a[i] > a[j]){
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
result.push (a[i]);
}
```

1) compare (if)
2) for loop - 1
3) swap (exchange)
   temp
   unwanted
   push
   empty [ ]
   wanted push
4) push
5) for loop - 2 → main

Object:

Syntax:

keys (always string) : value

```
var a = {name : "Raja", → comma
         Famil : 60 ; }
```

Slice: It remove the element from one position to another position.

return new array.

Ex: var a = [1, 2, 3, 4, 5, 6, 7, 8, 9].

a.slice(2, 6)

starting position.

Slice Method doesnot affect Original ar

Index Of: returns index position of a specified value. Index of returns -1 if the value is not found.

Ex: var a = [1, 2, 3, 4, 5, 6, 7]
(0 1 2 3 4 5 6)

a.indexOf(5);

Output:

Position: 4.

Includes:

It returns true if an array contain a specified value.

returns false if an array value not found

Ex: var a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
(0 1 2 3 4 5 6 7 8)

a.includes(2);

true

a.includes(2, 3);

false.

Sort: Sort an array as a ascending order.

But, It cheack only first position.

Ex: var a = [1, 2, 11, 23, 22, 4, 5]

a.sort();

Output: 1, 11, 2, 22, 23, 4, 5

Array Method:

* Push * pop * shift * unshift * slice * splice
* index of * Includes * Sort Concat.

Push():

* Add one (or) more Element at the end of array.

Ex:
```
var a = [1, 3, 7, 9, 10, 12];
a.length = 6;
a.push (25);
console.log (a);
```

Output:
1, 3, 7, 9, 10, 12, 25

concat():

* (Concatenates joins) two (or) more arrays. return: doesn't change original array. Create new array. (ending)

pop:

* It remove one element at the end of array.

Ex:
```
var a = [1,2,3,4,5]
a.pop(); (or) a.pop(5);
console.log(a);
```

Output:
1,2,3,4

* return b=5;

shift:

shift Method return: remove the 1st element of an array. return shift element. a.shift();

unshift:

return: length.

unshift Method add one (or) more element begining of an array. a.unshift (20, 30);

Splice:

* add (or) removes at any position.
* Overwrite the Original array.

return: remove item

Ex: a.splice (2, 1, "ARTA"); (position) index of

for Example:

```
a = [ { id : 1,
        name : Pen,
        Rs : 10,
        image : "image/1.jpg", },

      { id : 2,
        name : pen,
        Rs : 20, },

      { id : 3,
        name : pen,
        Rs : 25, },

      { id : 4
        name : Pencil,
        Rs : 2, },

      { id : 5,
        name : Pencil,
        Rs : 10, } ]
```

```
var output = a.filter (function (e){ return e.name
                                      = "pen";
console.log (output);
```

```
a = ["mango", "Apple", "banana"]
var output = a.map ( function (e){
                    return e.toUpperCase()})
console.log (output);
```

output: [MANGO, APPLE, BANANA].

→ map convert all the values in the array into various form.

→ it return a new array.

---

map in array of objects:

output: var a = a.map (function (e){ return e.pric

Result: pass;

Total() {
let sum = student.T + student.E + ... + student;
return sum;
} otherwise.

Total () { let sum = this.T + this.E + this.M + this.S + this.SS };
return sum; }.

output: console.log (student.name); console.log (student.Total());

Update:

student.english = 100;

For each () → used to take each value in the arra

Ex: a = [1, 10, 15, 17, 19, 90, 23, 30]
a.foreach (function (e) { if (e%2==0) {
console.log (e)}})

for.each () → does not return.
map () → it returns new array.
filter () → it returns new array.

filter ()

It is an array. and it give the new arr

Ex: a = [1, 10, 15, 17, 19, 90, 23, 230].

var b = a.filter (function (e) { return e%2==
});
console.log (b);

output: [10, 90, 230].

Ex: product = { name: "Apple",
color { red: 100,
black: 500 } }

}

Handling Method:

Syntax: Object name . key.

Ex: console.log (product . name);
(or)
console.log (product [name]);
(or)
console.log (product. color. red.);

capital ← Object Method:

it returns key & value.

* Object. keys * Object. Values * Object. Entries

All keys in array.

This Methods return (array)*

Ex: 1) Object. keys (product); ⇒ [name, color]

2) Object. values (product); ⇒ [Apple, { red: 100, black: 500 }].

3) Object. entries (product); ⇒ [[name, Apple], [color, { red: 100, black: 500 }]]

For Example: Var Student = { name: "APTA",
Roll.NO: 47,
T: 99,
E: 88,
M: 100,
S: 100,
S.S: 95,

Task! Change color: Get Method!

Input background color : [red]

Body background color : [yellow]

DOM

```
<section>
    <h1>.... </h1>
    <label>.... </label>.
    <input id="color" type="text", onchange="colorchange()">
</section>.
```

```
var colorname = document.getElementById("color")

function colorchange() {
    colorname.style.backgroundColor = colorname.value }
```

Color - different - Change

```
var a = ["red", "green", "blue", "purple"]
var number = (math.round (math.random()*a.length
document.body.style.backgroundcolor = a[number];
```

~~document.getElementsByClass("para").addEvent~~

Create Method:

1) document.CreateElement ("tagname");
2) setAttribute ("Attribute name", "Attribute value");
3) ClassList.add ("Class name");
   Remove
   toggle, Contains

else {arr3.push(arr[i]);}}

console.log(arr2);

Document Object Model

HTML element – Object-ಆ change ಮಾಡಬಹುದು.

Join – ಮಾಡಬಹುದು: ← get element → convert ca

* document.getElementsByName("p");
* document.getElementById("class Id name");

document.getElementsByClassName("class name");

Ex:

Var color name = document.getElementById("color");

Events:

* Click
* Focus
* mouseIn
* Changes
* load
* mouseout

Event Attributes:

* onclick
* onfocus
* onmouseIn
* onchange
* onload
* Onmouse Out:

Java script: (In line - style ಬರೆಯಬಹುದು)

* get ಮಾಡಬಹುದು ಮಾಡಬಹುದು.
* Element Create ಮಾಡಬಹುದು.
* Remove ಮಾಡಬಹುದು.

Build in Method :

1) Math. floor (8.17) → 8

2) math. floor (8.67) → 8

   math. ceil (8.01) → 9

   math. ceil (8.65) → 9

3) math. round (8.15) → 8

   math. round (8.65) → 9.

4) math. random () → Change between 0 and 1

   math. random () * 6 → 0.1 , 1.1 , 5.9

5) math. abs → it change the negative value into the

6) math. sign (10) → 1

   math. sign (-10) → -1

7) math. sqrt (2) → 2^2 → $\sqrt{4} = 2^2 = 2$

   math. cbert (2) → 2^3 → 8

8) math. min (27, 9, 11, 1) → 1

   math. max (27, 9, 11, 1) → 27 .

---

duplicate :

```
var arr=[];
var n=prompt("Enter the value");
for(let i=0 ; i<n ; i++){
        arr[i]=parseInt(prompt("Enter the "+(i+1)
                +"value"));
var arr2=[];
var arr3=[];
arr1.sort();
  let duplicate;
      for(let i=0 ; i<arr.length ; i++){
          if(arr[i] !== duplicate){
              arr2.push(arr[i]);
              duplicate=arr[i];
```

String Methods:

1) String.length() - return length of specified string. empty string is zero. a="Hello" → a.length() = 5
2) String.slice() - extracts part of string. return extract part. a = Hello world → a.slice(3, 9) = lo wor
3) String.substring()
4) String.substr() - same as slice. not change the original string. begin at position (specified) and return specify no. of chars. a = Hello ⇒ a.substr(-1, 4) ⇒ ello
5) String.replace() - method returns a new string with the value(s) replaced. not not change original string. a.replace(4,5) → a = [1,2,3,5]
6) String.replaceAll() - returns a new string with all value replaced. Induced in 2021. does not work in Internet Explorer.
7) String.toUpperCase() - convert a string to upper case. a = "helloworld" → a.toUpperCase() → HELLOWORLD
8) String.toLowerCase() - conver a string to lower case.
9) String.concat() - join 2 (or) more strings. method returns a new string. does not change existing
10) String.trim() - method remove white spaces from both ends of a string. doesnot change original string. a = " Hello " → a.trim() = "Hello"
11) String.trimEnd() - remove white space at the end of the string. since 2019.
12) String.padStart() - pad a string from start. pad a string with another string (multiple times) until its reach a given length. a.padStart(4, "*") → a = "5" → ***5
13) String.charAt() - return the character at the specified index in a string. a = "Hello" ⇒ a.charAt(0) → H
14) String.charCodeAt() - returns the unicode of the character at a specified index (position) in a string. a = "Hello world" → a.charCodeAt(1) → 69
15) String.split()
16) String.fromCharCode() - Methode converts Unicode values to characters. static method of string objects. a = 65 → String.fromCharCode(65) → A

```
{ id = 3,
  image = "image/1...",
  title = "Lotus",
  content = "This is Lotus", },
{ id = 4,
  image = "image/...",
  title = "Jasmine",
  content = "This is Jasmine", }]
```

Creation:

1) Section create 2) create container 3) create row

4) Create col. detail.function (function (e) {

```
var col = document.createElement ("div");
    col.classList.add (col);
    row.appendChild (col);

var card = document.createElement ("div");
    card.classList.add ("card-card");
    col.appendChild (card);

var picture = document.createElement ("img");
    card.appendChild (image);
* picture.setAttribute ("src", e.image);

var heading = document.createElement ("h1");
    card.appendChild (heading);
* heading.innerHTML = e.title;

var para = document.createElement ("p");
    card.appendchild (para);
    para.innerHTML = e.content; });
```

4) append Child.

5) inner HTML (or) inner Text.

Ex: HTML

```
<section id="card">
</section>
```

Java Script

```
var cardsec = document.getElementId(
var container = document.createElement("div"
container.classList.add("container
cardsec.appendChild(container);
```

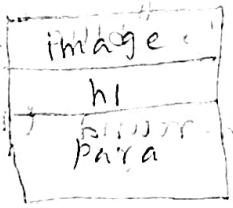
Image Add:

```
var image = document.createElement("img");
image.setAttribute("src", "....");
container.appendChild(image);
```

Content create:

```
var head = document.createElement("h1");
head.innerHTML = "Rose";
container.appedchild(head);
```

output: in browser:



Create Morethan One Card:

```
Let details = [ { id = 1,
    image = "image/....",
    title = "Rose",
    content = "This is Rose", },
  { id = 2,
    image = "image/....",
    title = "Lilly",
    content = "This is Lilly", },
```

Date Method

1) new Date()

* date objects are created with new Date().
* new Date() return a date object with current date & time

Ex:
```
var date = newDate();
console.log(date);
```

output:
Tue Aug 01 2023 20:05:53 GMT+ 0530 (India Standard Time).

---

2) getFullYear()

It returns the year of a date as a 4 digit number

ex:
```
var date = new Date();
date.getFullYear
console.log(a);
```

output:
2023

date.setFullYear(2029)

---

3) getMonth();

It return the month of a date as a number (0-11).

ex:
```
var date = newDate()
date.getmonth();
```

output:
7 (august)

---

4) getDate():

It return the day of a date as a number (1-31).

ex:
```
var date = newDate();
date.getDate();
```

output:
1

---

5) getDay(): It returns the weekday of a date as a number (0-6)

ex:
```
var date = newDate();
date.getDay();
```

output:
2

```
if (Ques.nextElementSibling.classList.contains("active")
    {
    ques.nextElementSibling.classList.remove("active");
    }
else {
    Ques.nextElement Sibling.classList.add("active");
    }
```

Tool-tip using icon - closing

```
var question = document.querySelectorAll(".ques");
var icons = document.querySelectorAll(".icon");
console.log(question);
question.forEach(function (e) {
    e.addEventListener("click", function () {
        if (e.nextElementSibling.classList.contains("hip")
            {
            e.nextElementSibling.classList.remove("hip");
        }
    });
});
icon.forEach(function (icon) {
    icon.addEventListener("click", function (e) {
        e.target.parentnode.classList.add("hip");
    })
})
```

10) getTime():

It returns the number of milliseconds since January 1 1970;

Ex:

```
var Date = new Date();
date.getTime();
```

output:

1690901974349,

String Search Method :

1) String . indexOf ()

2) String . lastIndexOf ()

3) String . search ()

4) String . match ()

5) String . matchAll ()

6) String . includes ()

7) String . startsWith ()

8) String . endsWith ()

Quary.

* addEventListener * quarySelector ("#id") * quarySelectorAll (".class")

* parentNode * nextElementSibling (previous)

For Example:

```
<div class="accord"> </div>
    <h1> Class = "qus" > Question1 </h1>
    <p> Answer1 </p> </div>

<div Class = "accord"> </div>
    <h1> Class = "qus" > Question2
    <p> Answer2 </p>
```

```
var accordQues = quesSelectorAll (".qus");
accordQues.foreach (function (Ques) {
    Ques.addEventListener("click", function() {
        Ques.nextElementSibling.classList.add
```

J.S Review

1) Data type - 4 2) array - 4 3) Math - 4

4) DOM - 3 5) Objects - 3 6) function - 2

7) Reserved key words - 4

1) Driver ID:

Input	condition	output
18	18>18	Eligible
21	21<18	not

var a = praseInt(prompt

a > 18

ID is Eligible

Id is not

2) +ve (or) -ve

var a = praseInt (prompt (" "));

I	Con	OS
-1	a<0	-ve
10	a>0	+ve

a > 0

positive

negative

3) Even (or) not:

In	Con	Out
12	a%2==0	Even.
3	a%2!=0	not even.

var a = praseInt(
prompt (" "));

a % 2 == 0

Even

4) Odd (or) not:

In	Con	Out
5	a%2!=	odd.
4	a%2==0	not odd.

var a = prase Int(
prompt (" ");

a % 2 != 0

Odd

not

Log in

```
var loginname = document.getElementById("logname");
var logpassword = document.getElementById("loginpasscoord);
var button = document.getElementById("loginnamebtn");

logbutton.addEventListener("click", function(){
        var a = loginname.value;
        var b = logpassword.value;
    var getusername = localStorage.getItem("regname");
    var getpassword = localStorage.getItem("reg password");
      if(a==getusername){
            if(b==getuserpassword){
                alert("success");}

            else{
                alert("unsuccess");}}

        else{
            alert("password incorrect");}})
```

Document Object Model:

```
var a = document.getElementsBytagname("hi");
```

array - ಆಗಿರುತ್ತದೆ.

6) getHours: It return the hours of a date as a number (0-23).

ex: var date = new Date();
date.getHours();

output:
20

7) getMinutes(): It return the minutes of a date as a number (0-59)

ex: var date = new Date();
date.getMinutes();

output:
24

8) getSeconds(): It return the seconds of a date as number (0-59)

ex: var date = new Date();
date.getSeconds();

output:
47

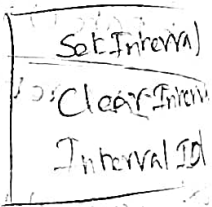
9) getMilliSeconds() It return the MillPseconds of a date as a number (0-999).

ex: var date = new Date();
date.getMilliSeconds();

output:
563

setInterval

Syntax:
setInterval(function(), 1000); → milliseconds → return id
clearInterval(Id)



FX:
var a = setInterval(function(){}, 1000);
clearInterval(a);

Register form:

var a = regname.value;
LocalStorage.setItem("regname", a);

var b = regpassword.value;
LocalStorage.setItem("regpassword", b);

Login.addEventListener("click", function(){
Location.href = "file://E://fabery...";
});

1) Month 1, 3, 5, 7, 8, 10, 12 → The number of days in month is 31.

month 4, 6, 9, 11 → The number of days in month is 30

month 2 → Leap year The number of days is 29.

not leap year days is 28.

Otherwise → not valied Month.

```
Let a = prompt ("Enter Month in number");
let b = prompt ("Enter the Year");

switch (true) {
case a==1 || a==3 || a==5 || a==7 || a==8 || a==10 || a==12
    : console.log ("Given Month - days is 31")
    break;
case a==4 || a==6 || a==9 || a==11
    console.log ("Given Month - days is 30")
    break;
case a==2 && b%4==0 :
    console.log ("Given Month - days is 29")
    break;
case a==2 && b%4!=0 :
    console.log ("not leap year : Given Month - days is 28")
    break;
default : console.log ("Enter valid month")}
```

Another Method:

Even:

```
Var a = prompt ("Enter the number")

if (a % 2 == 0)          else if (isNaN (a))          else

Even number              NaN (isNaN(num))             Even number
```

? : Conditional (or) Ternary condition:

Condition ? True statement : false statement

Switch Case:

```
Switch (condition (or) expression (or) Boolean) {
    case 1 : log ("Grade")
             break;
    case "2" : ____
             break;
    default : fail ; }.
```

For Example:

```
Let grade = prompt ("Grade");
switch (grade) {
    case "A" : log ("Grade A")
               break;
    case "B" : log ("Grade B")
               break;
    default : fail }
```

5) Leap (or) not:

In	Con	Out
2024	a%4==0	Leap
2025	a%4!=0	not leap.

```
var a= parseInt(
  prompt(" "));
a%4==0
Leap
not Leap
```

6) Divisible by 5.

In	Con	Out
15	a%5==0	divisible 5.
12	a%5!=0	not divisible 5.

```
var a= parseInt(
  prompt(" "));
a%5==0
Divisible 5
not Divsible 5.
```

7) Divisible by 7:

In	Con	Out
14	a%7==0	divisible 7.
8	a%7!=0	not divisible 7.

```
var a= parseInt(
  prompt(" "));
a%7==0
Divisible 7
not divisible 7.
```

8) Biggest and Small:

In	Con	Out
a=2, b=3	a<b	a is small, b is big
a=1, b=0	a>b	a is big, b is small.

```
var a = parse Int(
  prompt(" "));
var b = parseInt(
  prompt(" "));
a>b
a is big
b is small
a is smal
b is big
```

9) Greate than 10:

In	Con	Out
a=12	a>10	greater than 10
a=9	a<10	less than 10.

```
var a= parse Int(
  prompt(" "));
a>10
given no. is
greater than
10
Less
than
10.
```

12) Biggest number along 3 Number:

```
Let a = parseInt(prompt("1st"))
Let b = parseInt(prompt("2nd"))
Let c = parseInt(prompt("3rd"))

if (a>b && a>c){
        console.log("a is bigger")
        }
else if(b>a && b>c){
        console.log("b is bigger")
        }
else if (c>a && c>b){
        console.log("c is bigger")
        }
```

---

13) Calculate Total, percentage, Division. — Three

```
Let R.N = parseInt(prompt("Roll No"))
let Name = prompt("Name");
Let phy = parseInt(prompt("physics"))
let Chem = parseInt(prompt("Chemistry"))
let C.S = parseInt(prompt("Computer"));
var Total = phy + Chem + C.S;
var per = Total % 3;

if (per >= 90){
        console.log("Dis")
        }
else if (per >= 80){
        console.log("First")
        }
else if (per >=60){
        console.log("second")
        }
else if (per >=50) { console.log("Third")}
```

```
Let a = parseFloat(prompt("Enter height"));
if (a < 150) {
        console.log("The person is small");
    }
else if (a >= 150 && a < 165) {
        console.log("The person is Average height");
    }
else if (a >= 165 && a <= 195) {
        console.log("Taller")
    }
else { console.log("Abnormal height")
    }.
```

---

11) Eligibility: course eligible. Marks in Maths >= 65 and marks in phy >= 55 and Marks in Chem >= 50 and Total in 3 sub >= 190 (or) Total in Maths and phy >= 140

Input: phy = 65, Chem = 51, Maths = 72.
Total mark phy, Chem, Maths = 188.
total mark Math, phy = 187. → not eligible.

```
Let M = parseInt(prompt("Maths Mark")
Let P = parseInt(prompt("physics mark")
Let C = parseInt(prompt("chemistry Mark")
var Total = M+P+C;
var MP = M+P;
```

```
If (M >= 65 && P >= 50) {
    if (MP >= 140 || Total >= 190) {
```

```
else {
```

8) Check the given number is ~~equal~~ are not.

```
Let a = parseInt (prompt ("Enter 1-Value");
Let b = parseInt (prompt ("Enter 2-Value");
if (a == b) {
        console.log ("Equal")
        }
else if (a != b){
        console.log ("not equal")
else {
    console.log ("Enter only value")
    }.
```

---

9) Value of an integer m and display the value of n is 1 when m is larger than 0, 0 when m is and -1 when m is less than 0.

```
Let m = parseInt (prompt ("Enter m value");
~~Let n = parseInt (prompt ("Enter n value");~~

if (m>0){
    console.log("n value is 1")
    }

else if (m==0){
        console.log ("n value is 0")
        }

else if (m<0) {
        console.log ("n is -1")
        }

else {
        console.log ("please Enter the number")
        }.
```

---

10) Height less than 150.0 → The person is Small.
Height greater than or equal 150.0 and height less than 165.0 → The person is average height.
H- greater than (or) equal to 165.0 and Height less than (or equal) to 195.0 → Taller.
Otherwise → Abnormal height

```
let u = parseInt(prompt("Enter unit"));
let b = 15/100;
If (u <= 199) {
        var total = u*1.20;
        console.log(total);
        }
else if (u >= 200 && u < 400){
        var total = u*1.50;
    if (total >= 400){
        var tot = total/b;
      if (tot >= 100){
        console.log(tot+total);
        }
      else {
        console.log(total);
        }
      }
    else {
      console.log(total);
      }
  }
else if ((u >= 400) && (u < 600)){
        var total = u*1.80;
    if (total >= 400){
        var tot = total/b;
    if (tot >= 100) {
        console.log(tot+total);
        }
      else {console.log(total)
        }
    }
else 格{
    var total = u*2.00;
      var tot = total/b;
  if (tot >= 100) {console.log(tot+total);
        }
  else {console.log(total)}
```

```
else if (day == 7) {
        console.log (" Saturday")
    }.
else if {
    console.log (" please Enher the number
    }.
```

2) Vowel (or) Constant.

```
Let letter = prompt ("Enter a letter")
if ( letter == "a" || letter == "e" || letter == "i" || letter == "o" ||
                                    letter == "u") {
        console.log ("vowel letter")
else {
    console.log ("consonant");
    }.
```

3) Transaction : profit (or) loss

```
Let A = parseInt (prompt (" Debit Amount"))
Let B = parseInt (prompt ("Credit Amount")) else
if (a>b) {
        var c= a-b;
    console.log ("A - pay loss and B - profit")
    }
else if (a<b) {
        var c= b-a;
    console.log ("A - profit and B - loss")
    }
else {
    console.log ("no profit (or) loss")
    }.
```

4) upto 199 - 1.20 RS and, 200 and above but less than 400 - 1.50 RS, 400 and above but less than 600 - 1.80 RS, 600 and above - 2.00 RS

If bill exceed RS. 400 the surcharge of 15% will be charged and minimum bill should be RS 100/-

else { console.log ("fail") }.

14) Temparature:

```
Let a = praseInt (prompt ("Enter Temparature")
if (a<0) { console.log ("Freezing weather")}
else if (a>=10 && a<10) {console.log ("very cold weather")
else if (a>=10 && a<20) {console.log ("cold weather")
else if (a>=20 && a<30) {console.log ("Normal")
else if (a>=30 && a<40) {console.log ("Hot")
else { console.log ("very Hot") }.
```

---

11) day number in integer and display the day name in word format.

```
Let day = ParseInt (prompt ("Enter day")):
var days=["sunday",
If (day==1) {
    console.log ("sunday")
}
else if (day==2){
    console.log ("Monday")
}
else if (day==3){
    console.log ("Tuesday")
}.
else if (day==4){
    console.log ("Wednesday").
}
else if (day==5){
    console.log ("Thursday")
}
else if (day==6){
    console.log ("friday") }
```

Array. Position = length - 1

Let a = [1, "hello", true, undefined, null] → Structure format.
elements

filter:

Disadvantage - Not needed Memory - allocate.

Find - OR
Filter - AND

~~let~~ Array palindrome

Array Methods:

find:

It

foreach, map, filter, find, reduce, sort.

reduce

For Each:

To iterate value in an array and print the value.
It not return any value.

Syntax: forEach (callback function (value, index, arr

return value Position arr

Ex:

Ex:
```
num.forEach (function (value, index, array) {
    console.log (value, index, array)});
```

Map:

Not disturb Original value. It work as forEach.

Ex:

It return value (should be new array).

Syntax: array name.map (callback function (value, index, array))

Ex:
```
var res = num.map ( function (value, index, array){
    return value*2;
});
console.log (res)
```

return value*2 → give boolean value → (value=[1,2,4,5])

to avoid this, if (value*2 == 0)

```
while (a>0){
    b= a%.10;
    a= error parseInt (a/10);
    c= c*10+b;
}
if (c==rem){ console.log("palindrome");}
else { console.log("not a palindrome")}.
```

Prime - Number (Range)

```
Let S= parseInt (prompt ("Enter 1st Value"))
Let E= parseInt (prompt ("Enter 2nd Value"))
document.write ("prime number")

for(i=S; i<=E; i++){
    If (i==2) { console.log(i);
    }
    prime=true;
for(j=2; j<i ;j++){
    if (i%j==0){ prime=false;
                 break}.
    }
if (prime==true){
                 console.log(i);
    }
}
```

NOde:

```
<div>
    <h1>  → previous Sibling.
    <h2>  → Sibling
    <h3>  → Next Sibling.
</div>

Let parent = document.getElementById ("parent")
    console.log (parent.children[2])
    console.log (parent.firstElementChild);
    console.log (parent.lastElementChild);
```

loop

```
while (condition) {
        body of the loop
}
```

loop terminate - false

do while:

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

Palindrome:

while(a>0)

a=123. c=0

b = a%.10; a= a/10; c=c*10+b;

↓

a= parseInt(a/10)

Prime. Number:

```
Let a= parseInt (prompt("Enter number"));
for (i=2; i< a; i++){
        prime= true;
if (number %i ==0) {
        prime = false;
        break;
} }
if (prime==true) {
        Console.log (Palindrome prime)
}
else {
        console.log (Not prime);
}
```

Palindrome:

```
Let a= ParseInt (prompt ("Enter"))
Let rem = a;
var b;
var c=0;
```

7) let n = [29.76, 41.85, 46.5]

```
Let a = n.reduce (function (a,b){
    if (b>40){
        a.push (b)
    }, [])
return a
document.write (a);
```

---

x = 8, y = x++ diff: x = 8, y = ++x.

x++	++x
x = 9	x = 9
y = 8	y = 9.
first assign,	first Add
second Add	Next Assign.

{} → object represent **Object:** → Data Type (non-primitive).

object can store different Type of Data Type and Value. Collections of keys and Value.

Key - always String.

Value - String (or) Number (or) function (or) array.

Syntax: object →

```
const product = { name: ' ',
                  id : ' ',
                  price : ' ' }
```

Object literals:

→ dot , → square / Methods (ఆబ్జెక్ట్ మెథడ్). → Object document

Products.price (or) products['price']. - a print చేయాలి

But print అవ్వాలంటే: Javascript Object Notation

```
document.write (JSON.stringify (product))
```

Nested Object:

```
Let product = {
    name = 'apple',
    color {
        col1 : 'red'
        col2 : 'yellow' } }
```

Access

```
console.log (product.
    color.col1);
console.log (product
    ['color' ['col1']
```

1) Remove Negative Value: Let a = [-23, -20, -17, -12, -5, 0, 1, 5]

```
Let b= a. filter (function (e){
        return  e>=0
        })
document. write (b).
```

---

2) Let winner = ["Anna", "Beth", "cara"].

```
Let a=winners. filter (function (e){
        return e})
Let
document. write ("Gold" + a[0])
document. write ("Silver" + a[1])
document. write ("Bronze" + a[2]).
```

---

4) Circle: Let c = [10, 30, 50]

```
Let a = c. map (function (e){
    if (true){
        var b = Math. floor (e*e* 3.14)
        }
    return b
    })
document-write (a);
```

---

5) Let N= [10, 30, 50]

```
Let  a = N. reduce (function (a,b) {
        return a+b
        }
Let b  document. write (a /3)
```

---

6) Let N = [5, 10, 15, 20] → output [30, 40]

```
Let a = N. map (function (e) {
    if (e>0) {
        var b = e*2
        }
    return b }

document. write (a)
```

filter: It work as AND.

```
let res = num.filter(function(value){
    return value > 50;
});
console.log(res);
```

---

find: It work as OR. It give first satisfied value. It provide only one value.

---

reduce:

```
reduce(function(accumulator, currend value), initial value
```

zeroth position value — 1st position

Ex:

```
let num = [1,2,3].
let res = num2.reduce(function(a,b){
    return a+b;
});
console.log(res);
```

1st: a → 1, b → 2, a = 3

2nd: a = 3, b = 3, a = 6

Output: 6

Ex:

```
let res = num.reduce(function(a,b){
    return a+b;
},10)
console.log(res)
```

10, a → 1, b → 2, a = 11+2 = 13, 13+3 = 16.

Output: 16

Sort: It is only check ← Difference.

```
num = [1,10,20,5,2];
let res = num.sort(function(a,b){
    return a-b;
});
```

```
Name : "Mickey Spirit",
Genre : "Romance",
rating : 8.9,
Showday : "Mon - Tues"
Type : "Horror"
};

Phone = {
    brand { Type1 : "Vivo",
            Type2 : "Samsung",
            Type3 : "Mongi",
            Type4 : "Realme",
          },
    Color { C1 : "Red",
            C2 : "Black",
            C3 : "White",
            C4 : "Purple",
          },
    Storage : "64GB",
    price : 16000,
    Version : "Latest",
}
```

3) Let Travel = {

```
    Place : "Taj Mahal",
    country : "India",
    Transport : "Travel bus",
    Type : "7 wonder",
    Free : "Food & Photograph"};
```

Iteration:

```
Let car = [{
        brand : 'minivan',
        type : "Red",
        capacity : 7,
        },
        { brand : 'maruti'
          type : 'white',
          capacity : 7,
          },
        { brand : 'minivan'
          type : 'yellow'
          capacity : 4
        }].
```

Ex:

```
car.forEach (function (e) {
        console.log (e)
        console.log (e.brand)
        })
```

Filter:

```
Let a = car. filter (
        function (e) {
        return  e. capacity>
                })
Console.log (a)
```

↓ return key & valu

```
Let mark = {
        T : 99,
        E : 89,
        M : 100,
        S : 77,
        SS : 94,
    total () {
        Let sum = this.T + This.E + This.M +
                  this.S + This.SS

        return sum }

document.write ( ( mark. Total ())/5 ).
```

Extra Add:

```
product.name = 'banana'
product.quality = 40
```

output:

```
product = {"name": "banana",
           ....
           "quality": 40}
```

Let ca

function ⟷ object:

```
Let product = { name : "apple",
                color : "Red",
                price : 100,
                welcome() {
                    console.log ('hello')
                }
}
```

Call: product.welcome()

$ use

```
welcome() {
    Let msg = 'hi i am ${product.name} my price is
              ${product.price}.
```

Ex:

This - use: → use only object

```
welcome () {
    Let msg = 'hi I am ${this.name} my price is
              ${this.price}
```

Return- welcome() -ல store ஆகும்.

```
Let a = product.welcome()
console.log(a)
```

key இருக்கா இல்லையா பார்க்க has own property:

```
Let a = products.hasOwnProperty ('name')
if (a) {
    console.log ("yes it is here") }
else {
    console.log ("not it is nomore") }
```

otherwise console.log(a) → Return → Boolean Value.

return array:

Ex: Let person = { name: 'APTA'
age: 20,
clg: 'Sri' }

Let a = object.keys(person) → output ['name', 'age', 'clg']

Let a = object.values(person) → output ['APTA', '20', 'Sri'].

Let a = object.entries (person) → output [[name, APTA], [age, 20], [clg, Sri]]

For in:

objects — return key array — index (return.

For each and for of → syntax same as for in

array iterate பண்ணலாம். But objects iterate பண்ண முடியாது.

iteration: Best → for in (single object iterate).

	For in	For of	for each
array	return-index	return-value	return-all values
object	return-key	X	X

Ex:

```
Let phone = [{ name: 'vivo',
               price: 2000,
               color: 'yellow' },
             { name: 'oppo',
               price: 50,000,
               color: 'Black' }];
var phone1 = object.entries(Phone)
Let a = phone1.forEach(e) {
    for (e in Phone) {
        console.log(phone[e]) }};
```

array - object store பண்ணும்போது value + key

all

key's மட்டும் print

value

Instance of: → nested object இருக்கும்போது find பண்ணலாம்.

```
Let phone = [ {name: 'vivo',
               price: 2000,
               color: {
                   c1: 'Red',
                   c2: 'Yellow' } },
             {name: 'oppo',
              price: 50000,
              color: { c1: 'Black',
                       c2: 'Red' } }]
```

3) var library = [{ a : "Bill"
t : 'The Road Ahead',
reading status : true
},
{ a : 'steve',
t : 'waiter'
reading status : false }]

Output:
[Already read 'Bill' by the Road Ahead', You will need to read 'waiter' the fin book by 's

Let a = library . map (function (e, i) {
if (reading status == = true) {
document . write ('Already read ${e.a}' by the ${e.t}')}

else {
document.write ('you will need to read ${e.t} the final book by ${e.a}')}
});

Object - looping

* It is different to loop through an object in JS than arra looking over an array because JS objects are not obje iterable. unlike an array, a js object contains propert and value.
* when we iterate over an object, we can go through it properties, values or both, we can utilize different methods to handle different purpose.
* we will see all the methods you can use to loop over objects in JS with multiple examples.

Methods - loop:

const person = { name : 'APTA',
age : '25' }.

for (e in person) {
console.log (e, person[e]);
}

For in: return keys

syntax: for (varName in objName)

obj Name [varName].

console.log (e)
↓ keys

```
[illegible] 5[illegible]
[illegible] "[illegible]",
[illegible] "[illegible]",
Offer : "[illegible]"}
```

```
5) Let actor = {
        name : "kim Tae hyung"
        Model : "wkorea",
        band : "BTS",
        Award : "WWH",
        work : "Model -> kpop -> Actor"}
```

Task:

```
1) Let num = [{ f: 'mike',
                1: 'sheridan'},
              { f: 'Tim',
                1: 'Loe'},
              { f: 'John'
                1: Cate }]
```

output:

[Sheridan, Loe, Cate]

output:

['mike sheridan',
'Tim Loe',
'John Cate'].

```
Let b= num.map (function(e){
   var c = e.f + " " + e.1
       return c })
document.write (JSON.stringify (b));
```

```
2) Let user = [{ f: 'mike',
                 Loc: 'london'},
               { f: 'Tim'
                 Loc: 'us' }]
```

output:

['mike lives in
london'].

```
Let a = user.map (function (e) {
      var b = '${e.f} lives in ${e.location}'
          return b})
   document.write (JSON.stringify (a))
```

```
for (let i=0; i<size; i++){
        a[i]=ParseInt(prompt("Enter Value"))
    }
var dig=0
for (i=0; i<a.length; i++){
        b=a[i]
    var c=String(b)
    dig = dig+c.length
    }
Console.log(dig);
```

Divisible by 3, 4 & 5,

```
Let a=[]
let value=parseInt(prompt("Enter"))
for(i=0; i<value; i++){
        a[i]=parseInt(prompt("Enter "(i+1)"))
var d345=[]; var d34=[]; var d35=[], var d45=[],
var 3=[]; var 4=[], var 5=[].
  for (let i=0; i<value; i++){
      if (a[i]%3==0){
            if (a[i]%4==0){
                if(a[i]%5==0){
                    d3.push(a[i])
                    d4.push(a[i])
                    d5.push(a[i])
                    d345.push(a[i])
                }
            else{ d3.push(a[i])
                  d4.push(a[i])
                  d34.push(a[i])
                }
      }
  else if(a[i]%5==0){
                    d3.push(a[i])
                    d4.push(a[i])
                    d35.push(a[i])
  }
```

1) Design, Calculate The sum of n elements in a array and Average.

```
Let a = parseInt (prompt ("Enter the numbe of "))
Var sum = 0;
for (i=1; i<=a ;i++){
        sum = sum +i
}

var avg = sum/a;

console.log (sum);
console.log (avg);
```

---

2) Return how many even (or) odd number;

```
Let a = parseInt (prompt ("Enter the number"))
Var sum = 0.
for (i=1; i<=a; i++){
        if (i%.2==0) { sum = sum+i }
}.

consoledog (sum)
```

---

3) Factorial:

```
Let a = parseInt (prompt ("Enter"))

Let mul=1;
for (Let i =1; i<=a ; i++ ) { mul = mul *i }
console.log (mul);
```

---

4) cube Number (Range):

```
Var S= parseInt (prompt ("start"))
Var E= parseInt (prompt ("End"))

for (i =s; i<=e ; i++) {
        var cube = i**3
        document.write (i);
        document.write (cube); }.
```

---

5) Find digit in the array:

```
Var a=[]
Let size = parseInt (prompt ("Enter the numbes
```

```
for (let key in phone){
    if (phone[key] instance of 'object'){
        for (c in phone[key]){
            do.write ('${c} and ${phone[key][c]}<br>')
        }
    else {
        d.w ('${phone} and ${phone[key]}<br>')
```

Function - use করা যায় Nested object iterate

```
Let nest = function (e) {
    for (let key in e){
    if (e[key] instance of object){
        nest (e[key])
    else {
        do.write('${key} : ${ e[key]}<br>'}
    }}.
nest (phone);
```

Constructor: → like Date Date Method:

with new → object → initialize করা যায়।

Get → current Set → past/present/future

---

toLocaleString(): Let date = new Date(); D-M-Y

Date and time console.log (date.toLocaleString()); →

---

toLocaleDateString() → 31/08/2021 → ('de-De') → format-এ print করবে

toLocaleTimeString() → 12:52:27 ('us-Eg') → America

toJSON() → Y-M-D → console.log (date.toJSON()) →

split করা যায়

date.toJSON().slice(0,10)

↓ 2023-08-31

Syntax: Set Time out

```
Set TimeOut (hello, 3000);
function hello () { console.log ("hello world"); }
    Clear timeout ()
    Clear Interval ().
```

Syntax: Clear Time Out

```
ClearTimeOut (interval id)
Let retVel = set TimeOut (hello, 3000);
    clearTimeOut (retVel);
```

The Hello world is not printed. The timeout is cleared.

Program:

```
Let counter = 5
Let time = set TimeOut (count, 1000)
    function count () {
        if (counter > 0) {
            Clear TimeOut (time);
            console.log ("counter stop")
        }
    else { console.log ("counter start")
    }
```

Output: → counter started.

DOM - Document Object Model

Ex: name version.

DOM - Manipulation:

Ex:

```
Let para = document.CreateElement ("p")
para.innerHTML
para.setAttribute ("class", "para-head")
```

(name, 2 parameter → value. ↳ name : Value)

```
document.body.append (para);
para.style.color = "green"
para.ClassList.add ("para-cls")
```

(return HTML - collection ↓ array)

Create container

```
container name .append ("para")
```

query selector — return one

query selectorAll — return Array → return Node List.

(setInterval Id) ← clearInterval – clearTimeOut (set

<input type="button" onClick="stop()" value="stop

syntax:

function stop() {
clearInterval (res)
}

---

prompt - Input வாங்குதல் Alert - msg pass பண்ணுவது. syntax:

confirm - Yes / No

Open() (or) close()

open(link) → new tap-ல open ஆகும்.
close(link) → close.

History:

(Current)

forward(). - Next page open. previous-ல Next போவது.
go() - Back particular page.
back() - previous.

Ex: History.forward()
History.go(-2) (or) History.go(2)
↓ back ↓ forward
History.back().

---

navigator: → Broser detection → Ex: Version

Ex: navigator → All.
navigator.vendor → Google Inc (output)

Screen:

screen → availHeight: 720, availLeft: 0, ColorDepth: 24

Location:

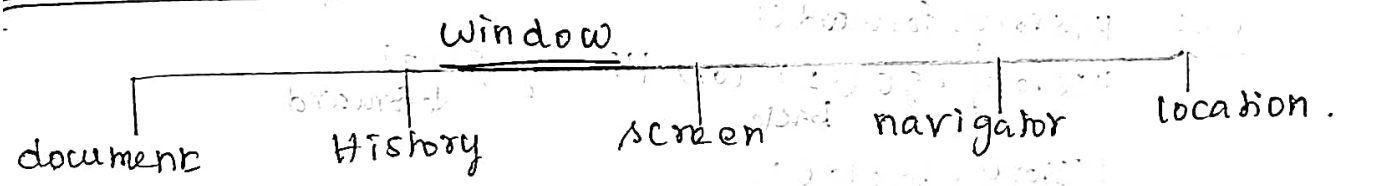
location → host - w3 school, href, pathname, hostname

```
        else { d3.push(a[i]) } }
    else if (a[i] % 4 == 0){
            if (a[i] % 5 == 0) {
                    d4.push(a[i])
                    d5.push(a[i])
                    d45.push(a[i])
                }
            else {
                d4.push(a[i])
                }
        }
    else { d5.push(a[i]) }
    }
console.log('divisible by 345 = [ ${d345} ] count is
                                    ${d345.length}')
```

BOM - used to Browser Interaction.

Browser object Model.

↓ default Object → window.



---

window object:

*Set Interval() *set time out() *alert()

but document. object

*prompt()

(object name

One time process ← **Set Time out:** → call back function.

Ex: OTP,

Syntax: Set Time out (function name, time mention)

↳ delay

For Example:

```
<input type="button" onclick="greet()", value="clic
function greet() {
var res = Set Time out (function() {
                alert ("welcome") }, 2000) }
```

log in:

```
let form= document.Selector("form");
    form.addEventLisher("submit", getData);

function getData(e){
        e.preventDefault()

let name= document.querySelector("#name").value.
let email= document.querySelector("#email").value.

let rname= LocalStorage.getItem("Rgname")
let remail= LocalStorage.getItem("Rgemail")

 if(name==rname && email==remail){
          alert("log in successfully")
        location.href="sample.html"
    }
 else{
    alert("enter correct username & email)
    }
```

---

like-set:

```
let heart= document.getElementById("heart")
let likeount= document.getElementById("Pa")

var a=203.
likeheart=true.
heart.addEventLisnner("click",function(){
    heart.addEventLisnner("click",function(e){
    }
if(likeheart==false){
    heart.style.color="black"
    likecount.innerHTML=(a)+"likes"
    likeheart=true.
}
else{heart.style.color="red"
    likecount.innerHTML=(a+1)+"likes"
    likeheart=false }})
```

(or)

```
Let but = document.getElementById("btn");
but.addEventListener("click", function() {
    let input1 = document.getElementById("text1").value;
    document.getElementById("t2").value = input1
```

---

refresh ஆகாமல் இருக்க. — PreEvent Default — Template literals → ``

```
Let form = document.querySelector("form")
form.addEventListener("submit", function(e){
    e.preventDefault();
```

Storage:

local - After save — எப்போதும் delete ஆகாது.

session - before save. — when window-close - it is delete.

---

setItem - save பண்ணுவது Storage-ல் set பண்ணுவது.

getItem - get the ~~store~~ data from local storage.

removeItem - remove பண்ணுவது.

Ex:
```
localStorage.setItem("flower", "lilly") → ("keyname", "value")
Let res = localStorage.getItem("flower")
console.log(res) → output: lilly.
localStorage.removeItem("flower")
```

Example: Register

```
Let form = document.querySelector("#form")
form.addEventListener("submit", storeData)
function storeData(e) {
    e.preventDefault()
    Let name = document.querySelector("#name").value;
    Let email = document.querySelector("#email").value;
    localStorage.setItem("Rgname", name)
    localStorage.setItem("Rgemail", email) }
```

addEventListner:

```
<body>
    <ul id=List>
    </ul>
    <button id="btn"> Generate </button>.
</body>
<script>
    Let food = ["idly", "Egg", "Dosa", "Noodles"]
Let ullist = document.getElementId("List")
  console.log(ul list)
Let but = document.getElement by Id("btn")
but.addEventlistner("click", process)
function process() { food.forEACH (function(e){
    Let list = document.createElement("li")
        ullist.appendChild(List);
        List.innerHTML = e;
    }
}
```

---

form-Validations! → user information get ചെയ്യാൻ.

Click to Move:

```
<input type="text" id="t1" >
<input type="text" id="t2" >
<button id="btn"></button>
```

input.value → string → number

Script:

```
Let index1 = document.getElementById("t1")
Let index2 = document.getElementById("t2")
Let but = document.getElementById("btn
but.addEventListner("click", function(){
    index2.value = index1.value
```

```
for (i=1 ; i<=n; i++){
    for (j=n; j>=i ; j--){
        document.write("   ")
    }
    Let c=1
for (k=1 ; k<=i ; k++){
    document.write("c  ")
    c= c*(i-k)/k
    }
}
```

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

```
for (i=1 ; i<=n ; i++){
    if(i==1 || i==n){
        for (i=1; i<=n ;i++){
            document.write("*  ")
        }
    else {
    for (i=1 ; i<=n ; i++){
```

```
* * * *
*     *
*     *
* * * *
```

```
for (i=1 ; i<=n ; i++){
    document.write("*  ")
}
for (j=1; j<n-1 ; j++){
    if(j==n || for(k=1 ; k<=n; k++){
        if(k==1 || k==n){
            document.write("*  ")
        }
    else {
        document.write("   
         ")
    }}
```

```
* * * *
*     *
*     *
* * * *
```

```
function default () {
    Let colremove = queryselectorAll (".col")
    colremove. forEach (function (e){
        e. remove ()
    })
    item (food)
}
```

Replace:

```
Let a = "Rs. 20"
Let b = a. replace ("Rs.", " ").
console.log (a) → "20"
ParseInt (b) → output - 20
```

Pattern: **Output:**

Letter = 1½

Number = ~~2~~ 1

Letter:

```
Let a = prompt ("Enter the Value")
for (i = 0 ; i < a ; i++){
    for (Let j = a ; j > i ; j--){
        document. write (String. fromCharCode ((j-1)+65))
    }
    document. write ("<br>")
    for (let k = 0 ; k <= i ; k++){
        document. write ("  ")
    }
}
```

Number:

j loop:

```
document. write ((j-1) + " ")
```

k loop:

```
document. write ("  ")
```

Card filter.

```
Let cont = document.qnarySelector("..container")
Let cards = document.createelement("div")
cards.ClassList.add("row")
cont.appendchild(cards)
Item(input)
function Item(input1){
  input.forEach(function(e){
  Let div = document.createElement("div")
  div.classList.add("col")
  Let div2 = document.CreateElement("div")
    div2.classList.add("col-card")
    div.appendchild(div2)
  Let img = document.createElement("img")
      div.appendchild(img)
      img.src=e.imgurl
Let head = document
      img.src = e.url
      div.appendchild(img)
Let head = document.CreateElement("h2")
  head.innerHTML = e.name
  div2.appendChild(head)
Let para = document.createElement("div")
  para.inneHTML = e.place.
      div2.appendchild(para)
```

Filter:

```
function sorts(){
    Leb sortrake= food.sort(function(a,b){
    return b.rating-(a.rating
    })
    Let colremove = document.querySelectorAll(".col")
```

Duplicate

1) Let a = [1, 1, 2, 3, 3, 5, 5].

```
Let result = [];
for (let i=0; i<a.length; i++){
    if (result.length==0){
        result.push(arr[i])
    }
if (result.indexOf(arr[i])===-1){
    result.push(arr[i])
    }
}
console.log(result)
```

```
Let result =
[...new Set(a)]
console.log(result)
↓
[1,2,3,5].
```

Add 2 Number

```
Let add = (a,b)=>{
while (b!=0){
    Let c = a&b
    a = a^b
    b = c<<1
    }
return a.
}
console.log(add(10,2))
```

Subtract two

```
Let sub = (a,b)=>{
while (b!=0){
    Let c = (~a)&b
    a = a^b
    b = c<<1
    }
return a.
}
console.log(sub(2,3))
```

Multiply 2 Number

```
Let mul = (x,y)=>{
    if (y==0){
        return 0;
        }
    if (y>0)
     return (x+mul(x,y-1))
    if (y<0)
     return -mul(x,-y)}
console.log(mul(5,-11))
```

Armstrong Number:

```
Let value = praseInt (prompt("size"))
    Let a=[], Let b=[]
for (i=0; i< value ; i++) {
        a[i]= parseInt (prompt((i+1)))   "value"
    }
console.log (a)
for(i=0; i< value; i++) {
    var sum=0
    var changeString = String (a[i])
    var tem = a[i]
    var power = changeString. length.
for(j=0; j< power; j++){
    var c= tem[i] ** power.
        sum = sum +c.
    }
if(sum== tem){
        b.push(tem)
    }
}
console.log (b)
```

---

given number is string / number / integer

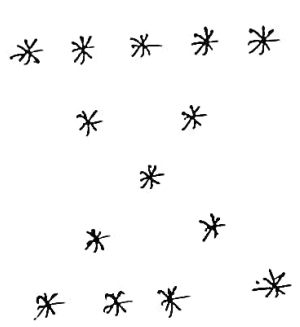
HTML

```
<input type="text" value="Enter the number" id="in">
```

JS:

```
Let input =document.getElementById("in").value.
    Atway_String.
Let a= math.floor (input)  -> 1.9 => 1
if (input>a && input<(a+1)) {
        log("integer") }
else if (input>0 || input<0){
        log(" number") }
else { log ("string") }
```

```
for (i=1 ; i<=n ; i++){
    document.write("* ")
}
```



```
Let a=parseInt (prompt("Enter"))
for (i=1; i<=a ; i++){
    document.write(" ",
}
document.write("<br>")

for (i=1; i<a-1 ;i++){
    for (j=1 ; j<=a ; j++){
        if ( j==i+1 || j==a-i){
            document.write("* ")
        }
        else {
            document.write("   ")
        }
    }
    document.write("<br>")
}
for (i=1; i<=a; i++){
    document.write("* ")
}
```

```
Let n=prompt("Enter the value
for (let i=1 ; i<=n ; i++){
    for (let j=n ; j>=i ; j-1
        document.write(" 
    }
    for (k=1 ; k<=i ; k++){
        write("* &nbs
    }
```