html
css
javascript

BY ATHARVA RAJ BHAGAT

hypertext markup language

HTML5

BY ATHARVA RAJ BHAGAT

HTML 5

```
<!DOCTYPE html>
<html>
<head> <title>
        </title>
</head>
<body>
    <h1>  </h1>
    <p>  </p>
    <hr>
    <br>
    <big>                    <blockquote>
    <small>                  <bdo dir="rtl">
    <i>                      <sub>
    <strong>                 <ins>
    <em>                     <mark>
    <sup>                    <del>
    <!--Comment-->
    <p  style="background-color : blue;"   =font-family   text-align
        style="color : red;">              =font-size

<header>          <nav>
    <main>        <article>        <section>
    <footer>                       <aside>

    <a href="  "  target="_blanks"> </a>

    <img src="  ">
    <img src="  "  alt="  "  height="  "  width="  " >

    <video src="  "></video>
    <video src="  "  controls  width="  "  poster="  " >
                  autoplay  loop
    <audio src="  "  controls>
```

Unordered list <ul> type (<ol type="a">)

ordered list <ol>

description list <dl> <dt> term
<dd> definition

<table> <caption> <thead> <tbody>
<tr> colspan=" "
<td> rowspan=" "
<th>

<table border=" " cellspacing=" " cellpadding=" ">

<Marquee>

 block format
 inline format

<input type="text" value=" "> autofocus autocomplete readonly size placeholder disabled maxlength required
<input type="password"> <input type="color">
<input type="email"> <input type="number"> min max step.
<input type="date"> <input type="reset">
< type="range" min: max: <input type="minute">
< type="file" > <input type="url">
< type="checkbox" > <input type="file"> multiple.
< type="radio" name=" "> <input type="image">
< type="submit" > <textarea rows=" " columns=" "></textarea>
<form></form>

embed youtube videos, pdf
<iframe src=" " frameborder=" " width=" "></iframe>
facebook graph tag twitter card tag

<Head> <meta charset="UTF-8"> <meta name="robots" content="index, follow">
<meta name="description" content=" ">
<meta name="author" content=" ">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="keywords" content=" ">
<link rel="canonical" href=" ">

email-link Contact email

image map <img src = " " usemap = " work " >
<map name = " work " >
<area shape = " rect " coords = " x,x , x,x " href = " " >
</map>
use image map generator from google to get cordinates or direct above format.

<form method = " GET / POST " action = " " > backend.
<form enctype = " multipart / form-data " >
<input type = " " name = " " >

dropdown:- <select> multiple
<option value = " " selected > " " </option>
<optgroup label = " " > </optgroup>

<input type = " radio " name = " " value = " " > checked >

<input type = " text " list = " search " >
<datalist id = " search " >
<option value = " " >
</datalist>

<label for = " " > </label>
<input type = " text " name = " " id = " " >

<fieldset>
<legend>

favicon <link rel = " icon " href = " " >
favicon generator

<html dir = " rtl " >
right to left align
<html lang = " " >

w3c validator - check html,

Cascading Style Sheets

CSS

BY ATHARVA RAJ BHAGAT

CASCADING STYLE SHEETS - CSS

style.css

in html

```
<head>
  <link rel="stylesheet" type="text/css" href="style.css">
```

external

internal

inline

Selectors

- element selector

```
p {
  color: red;
}
```

all paragraph elements become red

- id selector

```
<p id="p3">
#p3{
}
```

in html give <p> as <p id> for particular paragraph

unique id

- class selector

```
.green-text{
}
<p class="green-text">
```

can give multiple classes in a tag

Comment

```
/* comment */
```

Colors - htmlcolorcodes.com

Names

RGB — 0 to 255

RGBA — transperancy, alpha 0 to 1

HEX — #rrggbb

HSL — hue - 0 to 360, saturation - 0 to 100, lightness - 0 to 100

HSLA — alpha

Background

```
background-color:
background-image: url( );
background-position: right | left | center | top | bottom | center | px px | % %
background-repeat: no-repeat; repeat-x; repeat-y;
background-size: % % | px px | cover | contain
background-attachment: fixed; scroll;
```

// Multiple background

background-image: url(' '), url(' ');
background-position: , ;

// Border

border-style: solid dashed doubled groove outset inset ridge

All props. top right bottom left
(multiple styles for different side)

border-width: px
border-color :
border : 5px solid green
border-top-style :
width
color
border-radius: px

// Height and width

height : px %
width :
max-height : min-height :
max-width : min-height :

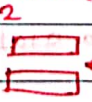
// Padding - text and gap

padding-top : px %
padding-bottom :
padding-left :
padding-right :
padding : px px / px px px px / px

create buttons

// Margin - gap between divs

margin-top : bottom left right
margin-auto :
margin collapse in top-bottom :- elm1 margin bottom: 30px, elm2 top : 50px. Total gap 50 px
but in right left it is added

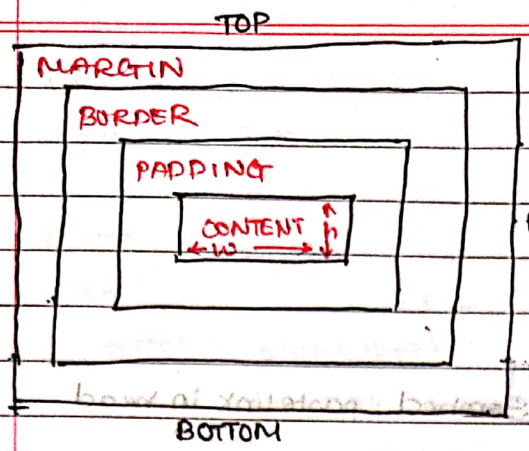


negative values allowed
create overlaps.

50px only (bigger value)

// BOX MODEL : When you set width and height of element, we just set it of content area. To calculate full size of element, we have to add padding, borders, margins.

total element width : width + left padding + right padding + left border + right + left margin + right
total element height : height + top + bottom +



← BOX MODEL

// TEXT

color:
text-align:
direction:
text-decoration: underline overline line-through
text-transform: uppercase lowercase Capitalize
line-height: px ← gap b/w lines.
text-indent: px
letter-spacing: ← gap b/w each character
word-spacing:
white-space:
word-wrap: break-word;
text-shadow: px ± px px colorname ;
(horizontal, vertical, blur)
writing-mode: vertical-lr rl

vertical-align: used in image to shift text if there is an image in the line.

// LINKS

```
a { text-decoration: none }
a: link {
    color: ;
}
a: visited {
}
a: hover {
}
a: active {
}
```

FONTS

font-size : px

font-weight : 300 ;

font-style : italic ;

font-variant : small-caps ;

online :- font-family : Google fonts :- embed : paste link in head

downloaded : @fontface {

font-family : abc ;

src : url ('fonts / Anton.ttf') ; }

p { font-family : abc ; }

h { font family : abc, poppins, etc. }

Font falls back ie. if a browser doesn't support any font or if font not found then the second mentioned font is used.

// CSS UNITS

Absolute	Relative
cm	em - relative to size of direct parent
mm	
In	rem - only relative to HTML tag
Px (1 inch = 96px)	Vh - 1% of viewport height
Pt (1 inch = 72pt)	Vw - 1% of viewport width (display window)
Pc (1 pc = 12pt)	

// CSS cursor

cursor : various cursors .

// CSS important

!important

recommended only in third-party code (if copied and don't want to change the code and apply ours as prior)

// BOX shadow

box-shadow : px px px px colorname inset ; ← for inside shadow

x ↑ y ↑ blur ↑ spread

// Opacity

opacity : 0 to 1

// Filter

```
img {
  filter : blur (10px)          contrast (%)
           brightness( %)       drop-shadow ( px px px color)
           grayscale (%)        saturate ( %)
           hue-rotate (90deg)   sepia ( %)
           opacity (%)
           invert (%)
```

// Image Sprite (divide mixed image) and keep showing at same portion of pic

background-image: ' ' height : px

background-position : 0px - desired px

Only available in background . Not in direct image.

// CSS Gradients

background-image : linear-gradient (green, blue)

(to right , colors)

(to bottom right , colors)

(120 deg , colors)

background-image :

radial-gradient

(circle, green 20%, blue 5%, etc)

default is ellipse .

// CSS overflow ← to control text inside container (block)

overflow-x : hidden ; default is visible !

overflow-y : scroll ; ← gives scroll bar for block on 'y' (vertical)

overflow : auto ;

// CSS Resize ← only in block and overflow needed !

resize : horizontal vertical both none

↑ can restrict textarea tag from expanding.

// LISTS

ul {
list-style-type: disc / circle / square ;

ol {
list-style-type : decimal-leading-zero / upper-roman / upper-alpha / lower-alpha / lower-roman

list-style-image: url(' ');
list-style-position: inside ;

// TABLES

table {
border :
border-collapse : collapse ; default is separate (cells are not joint)
border-spacing :
empty-cells : hide;
table-layout : auto ; fixed ;
width : px ;

th/td { height :
width :
padding :
text-align :
vertical-align :

caption { caption-side :

// CSS FUNCTIONS → calc() var()

calc (100% - 200px)

variables declaration syntax :-
--variable-name : value
declare variables inside
:root {
}
and these variables can be used in the function
var()

:root {
--primary-color : green
--btn-color : red
--text-color : blue
--variable-name ;
}

p {
background-color : var(--primary-color);

// BOX SIZING

box-sizing : content-box ← default (if padding is applied it is added on, content size is same box increases
border-box
↳ box size remains constant container size shrinks

// Initial & Inherit

initial → default — color: initial;

inherit → derive from parent tag — color: inherit;

// Object Fit

object-fit: contain fill cover

object-position:

// Pseudo-class ← state of a class

selector: pseudoclass { }

links: :active, :hover, :link, :visited

li : first-child { }

last-child { }

nth-child ( ) { odd, even, expression, no. }

nth-last-child (no)

inputs: :checked, :disabled, :enabled, :invalid, :read-only

input : focus { }

// Pseudo-element

selector :: pseudo-element { }

p :: first-line { }

first-letter { }

h1 :: before { } ← Add content using CSS

h1 :: after { content: " "; }

:: selection {

color: background:

}

// Display Properties

Block - div p h1 form — new line, full width — height & width

Inline - a img span — no new start, no full width — can't apply height & width

inline-blocks — no new start, no full width, height & width

display: block
inline
inline-block
none ← deletes space and element

visibility: hidden ← deletes element but not space.

// Position Properties

static: default

relative: previous posn — doesn't let anything to occupy space

absolute: relative to parent — allows to occupy, top: left:

fixed: bottom: right: ← fixed in viewport

// Layers & z-index

relative, absolute, fixed — top, left

z-index: 1, 2, 3 — priority of layer

// Float & Clear

float: right, left — adjust elements along with text

overflow: auto ← to box it in the container

Float makes element come out of container

clear: left, right, both ← not allow other elem to be on same line/space

Take divisions and create sidebars.

// 2D transform

a : hover {
display: in line -block;
tranoform: translate
scale
rotate (deg)
scalex y

a: {
display: inline-block
tranoform: skew
transform-origin: % %.

tranoform: matrix(scaleX() , skewY() , skewX() , scaleY() , translateX() , +Y());

// 3D transform

transform: perspective( ) rotate3d (x, y, z , deg)
translate (x, y, z)

perspective-origin:

// Transition Use with hover.

transition-property : width , background-color; /* all for all props.
transition-duration : sec
transition-delay :
transition-timing-function: ease ease-in-out (start, end middle speed)
transition: property , duration , timing-function , delay ;

// Animation

@keyframes changecolor {
from { property }
to { property }
} }

instead of from and to we can also use % of animation duration

0% { }
100% { }

0% { }
50% { }
100% { }

#animate {

properties
animation-name : changecolor ;
animation-duration: sec ;
animation-delay: sec ;
animation-iteration-count: no. of times / infinite
animation-direction: reverse
alternate
reverse-alternate
animation-timing-function:

// Print Styles — display website differently as original

```
@media print {
    p { font-size : 30px;}
    img { display : none }
    div { width:  }
}
```

but when we print the page we can give different layout for it.

// Responsive web Design → adapts to the device size

Viewport meta tag

```
<meta name = "viewport" content = "width: device-width,
                                   initial-scale : 1.0" >
```

Media Queries

```
@media screen and (max-width: px)
    #div {
        width : %
    }
    #div {
    }
    img {  width :
           height :  }
    video {  width :
             height :  }
```

// Advanced Selectors

- Attribute Selector ← to all attributes having same name

```
a[href="  "]{
}
```

- Universal selector ← everything in HTML used for debugging.

```
*{
}
```

- combinator selector

div a	descendant combinatory (child at any stage)	.red-p span{
div>a	child combinatory (direct child)	.red-p > span {
div+a	adjacent combinatory (right after sibling)	.red-p + h1 {
div~a	General sibling (all siblings)	.red-p ~ h1 {
h1/p	General sibling combinatory (right after)	.red-p, h1 {

// Specificity

! important will override everything
inline overrides everything except !important
External will work according to specificity.

} General Rule.

id selector has highest value
universal has lowest value

If same selector is written two or more times then one written at last is applied.

- Tricks: Point System

for combinations used as selector
e.g. div.red-span

10,000 to !important
1000 to inline
100 to id
10 to class, attribute or pseudo-class
1 for element selector & pseudo-element
0 to universal selector

// Multiple Columns

```
div{
  column-count:
  column-gap:
  column-rule-style:
  column-rule-width:
  column-rule-color:
  column-span:
  column-width:
```

column-rule: style width color

← for headers use in h1{ }

JavaScript

JS ES6

BY ATHARVA RAJ BHAGAT

JAVASCRIPT

// Variables

let variablename

typeof ← returns variable type

variable types in Javascript: String, number, boolean, undefined, null

// Displaying result

console.log( ) / log( )

document.write( )

// Objects

```
let person = {
    name: " ",
    age:
    };
```

```
let person = {
    name: ' ',
    age:  ,
    address: {
        street: ' ',
        city: ' '
    }
};
```

accessing object values

log(person.name)

person['name']

JSON.stringify(person)

Object.values(person)

// Arrays

let names = [' ', ' ', ' ', ' ', ' ']

let a = new Array( );

a = new Array();

a[0] =

a[1] =

accessing names[ ]

length names.length

.concat() .reverse() .join(" - "); delete a[0]

```
for (let n of names){
    log(n);
}
```

```
names.forEach(function(n, index){
    log(index + ' ' + n);
});
```

// Arithmetic operators + - * / %

// Comments // or /* */

// constants declaration → cannot reassign

const constantName

// Functions

```
function fname() {
    // define logic
    // log to print
}
```

```
function add(n1, n2){
    return n1+n2; or
    let sum = n1+n2;
    return sum;
}
add(10, 20);
log(add(10,20))
```

- toLowerCase();
- toUpperCase()
- endsWith()
- startsWith()
- keys()
- values()

// LOOPS

→ for(let i=0; i<=10; i++){ ← for loop

```
    // body
}
```

→ for(let n of names){ ← for n of

```
    //
}
```

→ names.forEach(){ ← forEach

```
    //
}
```

→ while(n < 5){ ← while

```
    //
}
```

→ do{ // ← do-while

```
} while()
```

// Comparison Operators == , < , <= , > , >= , !=

// Logical Operators && , || , !

// Type Coercion & === sign

number 0 == boolean false → true

string "0" == boolean false → true

number 1 == string "1" → true

even though the data types of 0 & false, "0" & false, 1 & "1" are different in the cases, the result is true because of type coercion. It tries to match data without seeing their data types.

To compare alongwith data type === operator is used.

0 === false → false

"0" === false → false

1 === "1" → false

// MAP | FILTER | REDUCE

MAP → transformations

```
let arr = [1,2,3,4,5].map(function(n){
        return n*2;
        });
                                    // 2,4,6,8,10
```

FILTER → filter list

```
let fil = [1,2,3,4,5,10,29,100].filter(function(n){
        return n%2==0;
        });
                                    // 2,4,10,100
```

REDUCE → adds the array / reduces it according to condition

```
let red = [1,2,3,4,5].reduce(function(accumulator,current){
        return accumulator + current;
        });
```

// CALLBACK

```
function callbackExample(name, callback){
    log(callback(name));
}
callbackExample("Atharva", function(name){
    return "Hello" +name;
    });
                                    // Hello Atharva
```

// Named Exports / Imports:

Creating two files. One main, second file where we keep all exports and methods.

Math.js

```
export let add = function (n1, n2){
    return n1 + n2;
    }
export let subtract = function (n1, n2) {
    return n1 - n2;
    }
```

Index.js

```
import * as Maths from './Math';
Maths.add(2,2);
Maths.subtract(4,2);
```

OR

```
import {add, subtract} from './Math';
add(2,2);
subtract(4,2);
```

// Default Export Import

Animal.js

```
export default class Animal{
    constructor(){
        console.log("I am animal");
        }
    getClassType(){
        return "Animal";
        }
    }
```

Index.js

```
import Animal from './Animal'
let animal = new Animal()
log(animal.getClassType());
```

// Template Literals Concatination

```
const name = 'Ath';
const country = 'Ind';
const age = 19;
log("Name: " + name + "country" + country + "age" + age);   ← normal way
log(`Name ${name} country ${country} age ${age}`);
```

// Spread Operator ... in Arrays

```
const arrayOne = ['A', 'B', 'C'];
const arrayTwo = ['X', 'Y', 'Z'];

const concatArray = [...arrayOne, ...arrayTwo];   // A B C X Y Z
```

```
const name = 'Atharva';
const nameToArray = [...name];
nameToArray.forEach(function(letter){
    log(letter);
});
```

// A T H A R V A

```
const numbers = [1, 5, 19];
const addNum = function (n1, n2, n3) {
        return n1+n2+n3;
    }
const addition = addNum(numbers[0], numbers[1], numbers[2]);   ← normal way
const addition = addNum(...numbers);   ← spread operator
```

// Spread in Objects.

```
const address = {
    city:
    country:
};
const name = {
    firstN:
    lastN:
};
const person = {...address, ...name};
```

```
const person = {
    city: address.city,
    country: address.country,
    firstN: name.firstN,
    lastN: name.lastN,
};
```

Normal concat of objects

// Arrow Functions =>

normal

```
const hello = function(){
  const es6 = 'ES6';
  return `Hello ${es6}`;
};

const add = function(n1,n2){
  return n1+n2;
};

const milesToKm = function(miles){
  return miles * 1.60;
};

log(hello());
log(add(100,100));
log(milesToKm(100));
```

arrow fn.

```
const hello = () => {
  const es6 = 'ES6';
  return `Hello ${es6}`;
};

const add = (n1,n2) => n1+n2;

const milesToKm = miles => miles*1.60;
```

// Lexical this.

If context of this is lost then use arrow function

```
const person = {
  name : 'Alex',
  cars : ['ferrari', 'lambo'],
  toString : function(){
    this.cars.forEach(function(car){
      log(`${this.name} has ${car}`);
    });
  }
}
```

← will not work.

change:

```
toString : function(){
  this.cars.forEach((car)=>{
    log(`${this.name} has ${car}`);
  });
```

// Enhanced Object properties

```
const calculator = (name) => {
    return {
        name: name,
        add: function (n1, n2) {
            return n1 + n2;
        }
    }
}
```

normal

enhanced.

```
name
add (n1, n2) {
    return n1+n2;
}
```

// Array Destructuring

```
const names = ['Anna', 'Mariam', 'Joe', 'Mark', 'Matt'];
const [a, b, c] = names;
log(`${a} ${b}`);                    // Anna Mariam Joe
const [a, , b, ...restofNames] = names;
log(`${a} ${b} ${restofNames}`)      // Anna Joe Mark,Matt
```

// Object Destructuring

```
const getUser = () => {
    return {
        name :
        surname :
        address : {
            country :
            city :
            fulladdress: {
                door;
            }
        }
    }
}
const user = getUser();
const {name, surname, address: {country: theCo}} = user;
log(name)
(surname)
[theCo]
```

normal

```
const name = user.name
const surn = user.surname
const co = user.address.country
```

// Classes :

```
class Animal {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
    eat() {
        log(`${this.name} is eating`);
    }
    sleep() {
        log(`${this.name} is sleeping`);
    }
    isAge() {
        log(`${this.name} is ${this.age} year old`);
    }
}
const tom = new Animal("tom", 2);
tom.eat();
tom.sleep();
tom.isAge();
```

// Inheritance

```
class Dog extends Animal {
    constructor(name, age, breed) {
        super(name, age);
        this.breed = breed;
    }
    breed() {
        log(`${this.name} is a ${this.breed}`)
    calcAge() {
        super.isAge();
    }
}
```

```
const mike = new Dog
mike.breed();
mike.eat();
mike.sleep();
mike.calcAge();
```

// Static Method

```
class Animal {
    static iAmStaticMethod() {
        log('I am static method in Animal');
    }
}

Animal.iAmStaticMethod();
```

// Export and Import

```
export class Classname{
}
```

```
import{Classname} from "./Filename";
```

// Popups

alert(); confirm(''); prompt(); → gather info (has textfield)

// Events → when an user performs an action

- onmouseover → places cursor on the target element
- onmouseout → removes cursor from the element

e.g.

```
<body>
  <p onmouseover="handleMouseOver(this)" onmouseout="handleMouseOut(this)">
    Mouse Example </p>
</body>
<script>
  function handleMouseOver(x){
    x.style.color = "red";
    x.innerHTML = "";
  }
  function handleMouseOut(x){
    x.style.color = "yellow";
  }
```

this → points to current object

→ to change contents.

- onclick → when a button is clicked

```
<Form name="calcform">
  Num1 :<input type="text" name:n1>
  Num2 :<input type='text' name:n2'>
  Result: <input text result>
  <input button Add value:Add
    onclick="calculate()">
</Form>
<script>
function calculate(){
  let x = parseInt(document.calcform.num1.value
  y =
  z = document.calcform.result
  x.value=x+y</i>
```

• onchange → after selecting an option

```
<select onchange="handleOnchange(this.value)">
  <option value:""> Select color </option>
  <option blue>Blue</>
  <green> </>
</select>
<script>
  function handleOnchange(a) {
    document.bgColor = x;
  }
```

window.location → changing website in option

document.body.style.backgroundImage = "url(" + x + ")"

window.print

• onfocus → places cursor
• onblur → removes and clicks somewhere else
• onkeyup → key removed (idle)
• onkeydown → when somekey is clicked
• onkeypress → when key is pressed
• onload → as soon as document loads
• setTimeOut(functionname, miliseconds) → delay
• onsubmit → when submit button is clicked

// Built-In Objects

Strings
- length
- indexOf()
- lastIndexOf()
- toLowerCase()
- toUpperCase()
- replace

Booleans

0 "" NaN null → false

"text" 1 → true

Math
- Math.sqrt()
- Math.PI
- pow()
- floor()
- ceil()
- round()

Date
- toLocaleTimeString()
- toLocaleDateString()
- toLocaleString()
- getHours() getSeconds()
- getMinutes()

setInterval

setInterval (function_name, seconds)

// Error Handling

```
try {
  //body
} catch(obj){
    //what to display.
} finally { }
```

obj.message → to show what error is on screen.

Custom errors :- throw

// Debugging

Inspect → Sources add Breakpoints to check errors.

// DOM - Document Object Model

document.getElementById(" ") →accessing HTML attribute.

- → .align
- → .style. color / bgcolor
- → .src
- etc.

Event Listener

```
<script>
  function handleEvent(){
      document.getElementById("button1").addEventListener("click",
                                                          handleB);
  function handleB(){
      alert(" ");
  }
}
window.addEventListener("load", handleEvent);
</script>

<body>
  <input type="button" value="click me" id="button1"/>
```

Create and remove node.

```
<script>
  function handleRemove(){
    let parent = document.getElementById("div1");
    let child = document.getElementById("p1");
    parent.removeChild(child);
    }
</script>
<body>
  <div id="div1">
    <p id="p1"> Text </p>
    <p id="p2"> Text </p>
  </div>
  <input type="button" onclick="handleRemove()">
</body>

function handleCreate(){
  let parent = document.getElementById("div1");
  let child = document.createElement("p");
  let text = document.createTextNode("Any Text");
  child.appendChild(text);
  parent.appendChild(child);
}
```

// BOM - Browser Object Model

screen	window.screen.width .height	→ screen properties
history	window.history.forward() .back()	→ go back and forward using manual buttons
navigator	navigator.appName .appCodeName .cookieEnabled .javaEnabled	→ browser properties.

// Form Validations

```
<body>
<form name="myform" action="registration.jsp" onsubmit="return validate()">
    Email: <input type="email" name="email">
    Password: <input type="password" name="pass">
    <input type="submit" value="register">
</form>
</body>
<script>
  function validate() {
    let email = document.myform.email.value;
    let pass = document.myform.pass.value;
    if (email=="") {
        alert("Email mandatory");
        document.myform.email.focus();
        return false;  ← form will not be submitted.
    } else if (password=="") {
        alert("Password mandatory");
        document.myform.pass.focus();
        return false;
    } else {
        alert("validation successful")
        return true;
    }
  }
</script>
```

if (isNaN(x)) ← NaN :- not a number

select .value ← for dropdowns

checkbox .checked ← for checkboxes getElementById( ).checked

radio .checked if(x[i].checked){
    valid=true;
    break;
}

// Regular expressions for validations

let pattern = /pattern/attributes;

= new RegExp(patten, attributes);

pattern.test(" ");

SYNTAX:

Brackets

[...]	any one character in brackets
[^...]	not in brackets
[0-9]	matches any decimal digit from 0 to 9
[a-z]	character from lowercase a to lc z
[A-Z]	uppercase A to uc Z

Quantifiers

x+	matches any string containing one or more x's
x*	zero or more x's
x{N}	sequence of N x's
x{3,5}	range of x's
x$	x at end of string
^x	x at beginning of string

Metacharacters

.	a single character
\s	a whitespace character (space, tab, newline)
\S	non-white space character
\d	a digit (0-9)
\D	a non-digit
\w	a word character (a-z, A-Z, 0-9, _)

```
<script>
function validate(){
    let exp = /^[A-Za-z]+$/   <- Alpha validation
    let user = document.myform.uname.value;
    let result = exp.test(user);
```

(^ should start, $ should end)

/^[A-Za-z0-9\s]+$/ /^[A-Za-z0-9\s]{3,10}$/