Font

There are many properties related to the font, such as the face, weight, style, etc. These properties allow you to change the style or complete look of your text.

Font-Family

```
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
```

Font-Style

```
font-style: italic;
```

Font-Variant

```
font-variant: small-caps;
```

Font-Weight

```
font-weight: bold;
```

Font-Size

```
font-size: larger;
```

Font

```
font: style variant weight size family;
```

Text

Text properties allow one to manipulate alignment, spacing, decoration, indentation, etc., in the document.

Text-Align

```
text-align: justify;
```

Letter-Spacing

```
letter-spacing: .15em;
```

Text-Decoration

```
text-decoration: underline;
```

Word-Spacing

```
word-spacing: 0.25em;
```

Text-Transform

```
text-transform: uppercase;
```

Text-Indent

```
text-indent: 0.5cm;
```

Line-Height

```
line-height: normal;
```

Background

As the name suggests, these properties are related to background, i.e., you can change the color, image, position, size, etc., of the document.

Background-Image

```
background-image: url("Path");
```

Background-Position

```
background-position: right top;
```

Background-Size

```
background-size: cover;
```

Background-Repeat

```
background-repeat: no-repeat;
```

Background-Attachment

```
background-attachment: scroll;
```

Background-Color

```
background-color: fuchsia;
```

Background

```
background: color image repeat attachment position;
```

Border

Border properties are used to change the style, radius, color, etc., of buttons or other items of the document.

Border-Width

```
border-width: 5px;
```

Border-Style

```
border-style: solid;
```

Border-Color

```
border-color: aqua;
```

Border-Radius

```
border-radius: 15px;
```

Border

```
border: width style color;
```

Box Model

In laymen's terms, the CSS box model is a container that wraps around every HTML element. It consists of margins, borders, padding, and the actual content. It is used to create the design and layout of web pages.

Float

```
float: none;
```

Clear

```
clear: both;
```

Display

```
display: block;
```

Height

```
height: fit-content;
```

Width

```
width: auto;
```

Margin

```
margin: top right bottom left;
```

Padding

```
padding: top right bottom left;
```

Overflow

```
overflow: hidden;
```

Visibility

```
visibility: visible;
```

Colors

With the help of the color property, one can give color to text, shape, or any other object.

Color

```
color: cornsilk;
```

Opacity

```
opacity: 4;
```

Template Layout

Specifies the visual look of the content inside a template

Box-Align

```
box-align : start;
```

Box-Direction

```
box-direction : normal;
```

Box-Flex

```
box-flex : normal;
```

Box-Flex-Group

```
box-flex-group : 2;
```

Box-Orient

```
box-orient : inline;
```

Box-Pack

```
box-pack : justify;
```

Box-Sizing

```
box-sizing : margin-box;
```

max-width

```
max-width: 800px;
```

min-width

```
min-width: 500px;
```

max-height

```
max-height: 100px;
```

min-height

```
min-height: 80px;
```

Table

Table properties are used to give style to the tables in the document. You can change many things like border spacing, table layout, caption, etc.

Border-Collapse

```
border-collapse: separate;
```

Empty-Cells

```
empty-cells: show;
```

Border-Spacing

```
border-spacing: 2px;
```

Table-Layout

```
table-layout: auto;
```

Caption-Side

```
caption-side: bottom;
```

Columns

These properties are used explicitly with columns of the tables, and they are used to give the table an incredible look.

Column-Count

```
column-count : 10;
```

Column-Gap

```
column-gap : 5px;
```

Column-rule-width

```
column-rule-width : medium;
```

Column-rule-style

```
column-rule-style : dotted ;
```

Column-rule-color

```
column-rule-color : black;
```

Column-width

```
column-width : 10px;
```

Column-span

```
column-span : all;
```

List & Markers

List and marker properties are used to customize lists in the document.

List-style-type

```
list-style-type: square;
```

List-style-position

```
list-style-position : 20px;
```

List-style-image

```
list-style-image : url(image.gif);
```

Marker-offset

```
marker-offset : auto;
```

Animations

CSS animations allow one to animate transitions or other media files on the web page.

Animation-name

```
animation-name : myanimation;
```

Animation-duration

```
animation-duration : 10s;
```

Animation-timing-function

```
animation-timing-function : ease;
```

Animation-delay

```
animation-delay : 5ms;
```

Animation-iteration-count

```
animation-iteration-count : 3;
```

Animation-direction

```
animation-direction : normal;
```

Animation-play-state

```
animation-play-state : running;
```

Animation-fill-mode

```
animation-fill-mode : both;
```

Transitions

Transitions let you define the transition between two states of an element.

Transition-property

```
transition-property: none;
```

Transition-duration

```
transition-duration : 2s;
```

Transition-timing-function

```
transition-timing-function: ease-in-out;
```

Transition-delay

```
transition-delay : 20ms;
```

CSS Flexbox

Flexbox is a layout of CSS that lets you format HTML easily. Flexbox makes it simple to align items vertically and horizontally using rows and columns. Items will "flex" to different sizes to fill the space. And overall, it makes the responsive design more manageable.

Parent Properties (flex container)

display

```
display: flex;
```

flex-direction

```
flex-direction: row | row-reverse | column | column-reverse;
```

flex-wrap

```
flex-wrap: nowrap | wrap | wrap-reverse;
```

flex-flow

```
flex-flow: column wrap;
```

justify-content

```
justify-content: flex-start | flex-end | center | space-between | space-arou
```

align-items

```
align-items: stretch | flex-start | flex-end | center | baseline | first bas
```

align-content

```
align-content: flex-start | flex-end | center | space-between | space-around
```

Child Properties (flex items)

order

```
order: 5; /* default is 0 */
```

flex-grow

```
flex-grow: 4; /* default 0 */
```

flex-shrink

```
flex-shrink: 3; /* default 1 */
```

flex-basis

```
flex-basis: | auto; /* default auto */
```

flex shorthand

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

align-self

```
align-self: auto | flex-start | flex-end | center | baseline | stretch;
```

CSS Grid

Grid layout is a 2-Dimensional grid system to CSS that creates complex responsive web design layouts more easily and consistently across browsers.

Parent Properties (Grid container)

display

```
display: grid | inline-grid;
```

grid-template-columns

```
grid-template-columns: 12px 12px 12px;
```

grid-template-rows

```
grid-template-rows: 8px auto 12px;
```

grid-template

```
grid-template: none | <grid-template-rows> / <grid-template-columns>;
```

column-gap

```
column-gap: <line-size>;
```

row-gap

```
row-gap: <line-size>;
```

grid-column-gap

```
grid-column-gap: <line-size>;
```

grid-row-gap

```
grid-row-gap: <line-size>;
```

gap shorthand

```
gap: <grid-row-gap> <grid-column-gap>;
```

grid-gap shorthand

```
grid-gap: <grid-row-gap> <grid-column-gap>;
```

justify-items

```
justify-items: start | end | center | stretch;
```

align-items

```
align-items: start | end | center | stretch;
```

place-items

```
place-items: center;
```

justify-content

```
justify-content: start | end | center | stretch | space-around | space-betwe
```

align-content

```
align-content: start | end | center | stretch | space-around | space-between
```

place-content

```
place-content: <align-content> / <justify-content> ;
```

grid-auto-columns

```
grid-auto-columns: <track-size> ...;
```

grid-auto-rows

```
grid-auto-rows: <track-size> ...;
```

grid-auto-flow

```
grid-auto-flow: row | column | row dense | column dense;
```

Child Properties (Grid items)

grid-column-start

```
grid-column-start: <number> | <name> | span <number> | span <name> | auto;
```

grid-column-end

```
grid-column-end: <number> | <name> | span <number> | span <name> | auto;
```

grid-row-start

```
grid-row-start: <number> | <name> | span <number> | span <name> | auto;
```

grid-row-end

```
grid-row-end: <number> | <name> | span <number> | span <name> | auto;
```

grid-column shorthand

```
grid-column: <start-line> / <end-line> | <start-line> / span <value>;
```

grid-row shorthand

```
grid-row: <start-line> / <end-line> | <start-line> / span <value>;
```

grid-area

```
grid-area: <name> | <row-start> / <column-start> / <row-end> / <column-end>;
```

justify-self

```
justify-self: start | end | center | stretch;
```

align-self

```
align-self: start | end | center | stretch;
```

place-self

```
place-self: center;
```

JavaScript Basics

Set of JavaScript basic syntax to add, execute and write basic programming paradigms in Javascript

On Page Script

Adding internal JavaScript to HTML

```
<script type="text/javascript"> //JS code goes here </script>
```

External JS File

Adding external JavaScript to HTML

```
<script src="filename.js"></script>
```

Functions

JavaScript Function syntax

```
function nameOfFunction () {
// function body
}
```

DOM Element

Changing content of a DOM Element

```
document.getElementById("elementID").innerHTML = "Hello World!";
```

Output

This will print the value of a in JavaScript console

```
console.log(a);
```

Conditional Statements

Conditional statements are used to perform operations based on some conditions.

If Statement

The block of code to be executed, when the condition specified is true.

```
if (condition) {
// block of code to be executed if the condition is true
}
```

If-else Statement

If the condition for the if block is false, then the else block will be executed.

```
if (condition) {
// block of code to be executed if the condition is true
} else {
// block of code to be executed if the condition is false
}
```

Else-if Statement

A basic if-else ladder

```
if (condition1) {
// block of code to be executed if condition1 is true
} else if (condition2) {
// block of code to be executed if the condition1 is false and condition2 is
} else {
// block of code to be executed if the condition1 is false and condition2 is
}
```

Switch Statement

Switch case statement in JavaScript

```
switch(expression) {
case x:
// code block
break;
case y:
```

```
// code block
break;
default:
// code block
}
```

Iterative Statements (Loops)

Iterative statement facilitates programmer to execute any block of code lines repeatedly and can be controlled as per conditions added by the programmer.

For Loop

For loop syntax in javascript

```
for (statement 1; statement 2; statement 3) {
// code block to be executed
}
```

While Loop

Runs the code till the specified condition is true

```
while (condition) {
// code block to be executed
}
```

Do While Loop

A do while loop is executed at least once despite the condition being true or false

```
do {
// run this code in block
i++;
} while (condition);
```

Strings

The string is a sequence of characters that is used for storing and managing text data.

charAt method

Returns the character from the specified index.

```
str.charAt(3)
```

concat method

Joins two or more strings together.

```
str1.concat(str2)
```

index of method

Returns the index of the first occurrence of the specified character from the string else -1 if not found.

```
str.indexOf('substr')
```

match method

Searches a string for a match against a regular expression.

```
str.match(/(chapter \d+(\.\d)*)/i;)
```

replace method

Searches a string for a match against a specified string or char and returns a new string by replacing the specified values.

```
str1.replace(str2)
```

search method

Searches a string against a specified value.

```
str.search('term')
```

split method

Splits a string into an array consisting of substrings.

```
str.split('\n')
```

substring method

Returns a substring of a string containing characters from the specified indices.

```
str.substring(0,5)
```

Arrays

The array is a collection of data items of the same type. In simple terms, it is a variable that contains multiple values.

variable

Containers for storing data.

```
var fruit = ["element1", "element2", "element3"];
```

concat method

Joins two or more arrays together.

```
concat()
```

indexOf method

Returns the index of the specified item from the array.

```
indexOf()
```

join method

Converts the array elements to a string.

```
join()
```

pop method

Deletes the last element of the array.

```
pop()
```

reverse method

This method reverses the order of the array elements.

```
reverse()
```

sort method

Sorts the array elements in a specified manner.

```
sort()
```

toString method

Converts the array elements to a string.

```
toString()
```

valueOf method

returns the relevant Number Object holding the value of the argument passed

```
valueOf()
```

Number Methods

JS math and number objects provide several constant and methods to perform mathematical operations.

toExponential method

Converts a number to its exponential form.

```
toExponential()
```

toPrecision method

Formats a number into a specified length.

```
toPrecision()
```

toString method

Converts an object to a string

```
toString()
```

valueOf method

Returns the primitive value of a number.

```
valueOf()
```

Maths Methods

ceil method

Rounds a number upwards to the nearest integer, and returns the result

```
ceil(x)
```

exp method

Returns the value of E^x.

```
exp(x)
```

log method

Returns the logarithmic value of x.

```
log(x)
```

pow method

Returns the value of x to the power y.

```
pow(x,y)
```

random method

Returns a random number between 0 and 1.

```
random()
```

sqrt method

Returns the square root of a number x

```
sqrt(x)
```

Dates

Date object is used to get the year, month and day. It has methods to get and set day, month, year, hour, minute, and seconds.

Pulling Date from the Date object

Returns the date from the date object

```
getDate()
```

Pulling Day from the Date object

Returns the day from the date object

```
getDay()
```

Pulling Hours from the Date object

Returns the hours from the date object

```
getHours()
```

Pulling Minutes from the Date object

Returns the minutes from the date object

```
getMinutes()
```

Pulling Seconds from the Date object

Returns the seconds from the date object

```
getSeconds()
```

Pulling Time from the Date object

Returns the time from the date object

```
getTime()
```

Mouse Events

Any change in the state of an object is referred to as an Event. With the help of JS, you can handle events, i.e., how any specific HTML tag will work when the user does something.

click

Fired when an element is clicked

```
element.addEventListener('click', ()=>{
// Code to be executed when the event is fired
});
```

oncontextmenu

Fired when an element is right-clicked

```
element.addEventListener('contextmenu', ()=>{
// Code to be executed when the event is fired
});
```

dblclick

Fired when an element is double-clicked

```
element.addEventListener('dblclick', ()=>{
// Code to be executed when the event is fired
});
```

mouseenter

Fired when an element is entered by the mouse arrow

```
element.addEventListener('mouseenter', ()=>{
// Code to be executed when the event is fired
});
```

mouseleave

Fired when an element is exited by the mouse arrow

```
element.addEventListener('mouseleave', ()=>{
// Code to be executed when the event is fired
});
```

mousemove

Fired when the mouse is moved inside the element

```
element.addEventListener('mousemove', ()=>{
// Code to be executed when the event is fired
});
```

Keyboard Events

keydown

Fired when the user is pressing a key on the keyboard

```
element.addEventListener('keydown', ()=>{
// Code to be executed when the event is fired
});
```

keypress

Fired when the user presses the key on the keyboard

```
element.addEventListener('keypress', ()=>{
// Code to be executed when the event is fired
});
```

keyup

Fired when the user releases a key on the keyboard

```
element.addEventListener('keyup', ()=>{
// Code to be executed when the event is fired
});
```

Errors

Errors are thrown by the compiler or interpreter whenever they find any fault in the code, and it can be of any type like syntax error, run-time error, logical error, etc. JS provides some functions to handle the errors.

try and catch

Try the code block and execute catch when err is thrown

```
try {
Block of code to try
}
catch(err) {
Block of code to handle errors
}
```

Window Methods

Methods that are available from the window object

alert method

Used to alert something on the screen

```
alert()
```

blur method

The blur() method removes focus from the current window.

```
blur()
```

setInterval

Keeps executing code at a certain interval

```
setInterval(() => {
// Code to be executed
}, 1000);
```

setTimeout

Executes the code after a certain interval of time

```
setTimeout(() => {
// Code to be executed
}, 1000);
```

close

The Window. close() method closes the current window

```
window.close()
```

confirm

The window.confirm() instructs the browser to display a dialog with an optional message, and to wait until the user either confirms or cancels

```
window.confirm('Are you sure?')
```

open

Opens a new window

```
window.open("https://www.codewithharry.com");
```

prompt

Prompts the user with a text and takes a value. Second parameter is the default value

```
var name = prompt("What is your name?", "Harry");
```

scrollBy

```
window.scrollBy(100, 0); // Scroll 100px to the right
```

scrollTo

Scrolls the document to the specified coordinates.

```
window.scrollTo(500, 0); // Scroll to horizontal position 500
```

clearInterval

Clears the setInterval. var is the value returned by setInterval call

```
clearInterval(var)
```

clearTimeout

Clears the setTimeout. var is the value returned by setTimeout call

```
clearTimeout(var)
```

stop

Stops the further resource loading

```
stop()
```

Query/Get Elements

The browser creates a DOM (Document Object Model) whenever a web page is loaded, and with the help of HTML DOM, one can access and modify all the elements of the HTML document.

querySelector

Selector to select first matching element

```
document.querySelector('css-selectors')
```

querySelectorAll

A selector to select all matching elements

```
document.querySelectorAll('css-selectors', ...)
```

getElementsByTagName

Select elements by tag name

```
document.getElementsByTagName('element-name')
```

getElementsByClassName

Select elements by class name

```
document.getElementsByClassName('class-name')
```

Get Element by Id

Select an element by its id

```
document.getElementById('id')
```

Creating Elements

Create new elements in the DOM

createElement

Create a new element

```
document.createElement('div')
```

createTextNode

Create a new text node

```
document.createTextNode('some text here')
```