

Q1. Create a basic HTML document. (All basic tags along with required attributes, also mention the tags that goes in head element).

```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to Trellissoft</title>
</head>
<body>
    <h1>heyyy 1</h1>
    <h2>heyyy 2</h2>
    <h3>heyyy 3</h3>
    <h4>heyyy 4</h4>
    <h5>heyyy 5</h5>
    <h6>heyyy 6</h6>

    <p>This is a paragraph of text.</p>

    <p>This paragraph<br>has a line break.</p>
    <p>This is a comment <!-- This shows the extra information or
explanation about the code --></p>
</body>
</html>
```

heyyy 1

heyyy 2

heyyy 3

heyyy 4

heyyy 5

heyyy 6

This is a paragraph of text.

This paragraph
has a line break.

This is a comment

Q2. Mention 3 ways of styling a document, along with cascading order (mention priority using comments).

There are 3 types in styling the a document.

Inline styles which has the highest priority. Then Internal styles which has medium Priority. The external Styles which has the lowest priority.

```
<!DOCTYPE html>
<html>
<head>
  <title>Welcome to Trellissoft</title>
  <link rel="stylesheet" href="styles.css">
  <style>

    /* Priority: Medium */
    h2 {
```

```
        color: blue;
    }

</style>
</head>
<body>
    <h1 style="color: green;">heyyy 1</h1> <!-- Priority: Highest -->
    <h2>heyyy 2</h2>
    <h3>heyyy 3</h3>

    <p>This is a paragraph of text.</p>

    <p class="purple-text">This paragraph<br>has a line break.</p>
    <p>This is a comment <!-- This shows the extra information or
explanation about the code --></p>
</body>
</html>
```

← → ↻ ⓘ File | C:/Use

heyyy 1

heyyy 2

heyyy 3

This is a paragraph of text.

This paragraph
has a line break.

This is a comment

Q3. Explain CSS selectors: simple, combinator, pseudo-class(links/form/table), pseudo-elements and attribute selectors.

Selectors are used to target and precisely style certain elements. Different types of selectors exist.

-Simple selectors use the name, id, and class to choose the elements.

The id selector chooses a particular HTML element by using the id attribute.

HTML components having a specific class attribute are chosen using the class selector. Write a period (.) followed by the class name to pick elements that belong to that class

All HTML items on the page are selected by the global selector (*).

–All HTML elements that share the same style definitions are chosen by the grouping selector.

- A CSS selector may include multiple simple selectors.

In CSS, there are four distinct combinators:

1-selector for descendants (space)

2-selector for kids (>)

3-selector for adjacent siblings (+)

4-Selecting siblings generally ()

All elements that are descendants of a given element are matched by the descendant selector.

The child selector chooses all elements that are a specific element's offspring.

The adjacent sibling selector is used to choose an element that follows another particular element exactly.

The general sibling selector chooses every element that is a specified element's next sibling.

–The attribute selector is used to select elements with a specified attribute.

–Pseudo-classes are used to define a special state of an element. They are preceded by a colon (:) in CSS.

Some common pseudo-classes include hover, active, focus, and nth-child().

hover: Targets an element when the mouse pointer is over it.

active: Targets an element when it's being activated

focus: Targets an element when it receives focus

nth-child(): Targets elements based on their position within a parent element.

–Pseudo-elements are used to style a specific part of an element

. They are preceded by double colons (::) in CSS. Common pseudo-elements include ::before and ::after, which add content before or after an element's content.

::before: Inserts content before the content of the selected element.

::after: Inserts content after the content of the selected element.

Attribute selectors target elements based on the presence or value of attributes. They are enclosed in square brackets ([]) in CSS. Common attribute selectors include attribute and attribute=value

Q4. Mention ways to create an array, access and manipulate it. How to recognise if a variable is an array?

To recognise the different elements in the array we can use typeof method or instanceof() method. The isArray() method returns true if the given elements are in array and false if its not.

literal is the easiest way to create a JavaScript Array. the syntax is `const array_name = [item1, item2, ...,];`

–Accessing the First Array Element

```
<p id="fru"></p>
```

```
<script>
const fruits = ["Banana", "Orange"];
document.getElementById("fru").innerHTML = fruits[0];
</script>
```

Banana

Output –

–Accessing the Last Array Element

```
<p id="fru"></p>
```

```
<script>
const fruits = ["Banana", "Orange"];
document.getElementById("fru").innerHTML = fruits[fruits.length-1];
</script>
```

Orange

Output -

–You access an array element by referring to the index number.

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>
<p id="demo1"></p>
<p id="demo2"></p>

<script>
```

```

const names = ["Ram", "Sham", "Dev"];/** Creating an array */
document.getElementById("demo").innerHTML = names;
document.getElementById("demo1").innerHTML = names[1];/**Accessing Array
Elements*/

const names1 = new Array("ram", "sham", "dev");/** Creating an array using
js keyword*/
names1[0] = "raj";/** Changing an Array Element */
document.getElementById("demo2").innerHTML = names1;

</script>

</body>
</html>

```

Output -



Ram,Sham,Dev

Sham

raj,sham,dev

–The pop() method removes the last element from an array and the push() method adds a new element to an array at the end. The shift() method removes the first array element and "shifts" all other elements to a lower index.

<p id="d1"></p>

<p id="d2"></p>

```
<p id="d3"></p>
<p id="d4"></p>
```

```
<script>
const fruits = ["one", "two"];
document.getElementById("d1").innerHTML = fruits;
fruits.push("three");
document.getElementById("d2").innerHTML = fruits;
fruits.pop("three");
document.getElementById("d3").innerHTML = fruits;
fruits.shift();
document.getElementById("d4").innerHTML = fruits;
</script>
```

Output -

one,two

one,two,three

one,two

two

Q5. Find the size of an array, access the first and last element of array and add new element using appropriate array property.

–The length property returns the length (size) of an array.

```
<p id="d5"></p>
```

```
let size = fruits.length;
document.getElementById("d5").innerHTML = size;
```

Output -

two

1

–Assessing the first and the last element of the array

```
<p id="d6"></p>
```

```
<p id="d7"></p>
```

```
<script>
const fruits = ["one", "two", "four", "five"];
document.getElementById("d6").innerHTML = fruits[0];
```

```
document.getElementById("d7").innerHTML = fruits[fruits.length-1];  
</script>
```

Output -

one

five

–Push method is used to add an element to an array

```
<p id="d1"></p>
```

```
const fruits = ["one", "two", "four", "five"];  
fruits.push("three");
```

```
document.getElementById("d1").innerHTML = fruits;  
Output -
```

one,two,four,five,three

Q6. Mention different methods to add new element to an array.

–The push() method returns the new array length:

–The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:

```
<p id="d1"></p>  
<p id="d2"></p>  
<p id="d3"></p>  
<script>  
const fruits = ["one", "two", "four", "five"];  
document.getElementById("d1").innerHTML = fruits;  
fruits.push("three");  
document.getElementById("d2").innerHTML = fruits;  
fruits.unshift("eight");  
document.getElementById("d3").innerHTML = fruits;  
</script>
```

Output -

one,two,four,five

one,two,four,five,three

eight,one,two,four,five,three

–The concat() method creates a new array by merging (concatenating) existing arrays

```
<p id="demo"></p>
```

```
<p id="demo2"></p>
```

```
<script>
```

```
const myGirls = ["Cecilie", "Lone"];
```

```
const myBoys = ["Emil", "Tobias", "Linus"];
```

```
const myChildren = myGirls.concat(myBoys);
```

```
document.getElementById("demo").innerHTML = myChildren;
```

```
</script>
```

Output -

Cecilie,Lone,Emil,Tobias,Linus

–The splice() method can be used to add new items to an array

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
document.getElementById("demo1").innerHTML = fruits;
```

```
fruits.splice(2, 0, "Lemon", "Kiwi");
```

```
document.getElementById("demo2").innerHTML = fruits;
```

Output -

Banana,Orange,Apple,Mango

Banana,Orange,Lemon,Kiwi,Apple,Mango

Q7. Mention different methods to delete element from an array.

–splice() can be used to remove elements.

```
<p id="demo1"></p>
```

```
<p id="demo2"></p>
```

```
<script>  
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo1").innerHTML = fruits;  
fruits.splice(0, 1);  
document.getElementById("demo2").innerHTML = fruits;
```

```
</script>
```

Output -

Banana,Orange,Apple,Mango

Orange,Apple,Mango

–The pop() method removes the last element from an array

```
<script>  
const fruits = ["one", "two","four","five"];  
document.getElementById("d2").innerHTML = fruits;  
fruits.pop("five");  
document.getElementById("d3").innerHTML = fruits;
```

```
</script>
```

Output -

one,two,four,five

one,two,four

–The shift() method returns the value that was "shifted out"

```
<script>  
const fruits = ["one", "two","four","five"];  
document.getElementById("d2").innerHTML = fruits;  
fruits.shift();  
document.getElementById("d3").innerHTML = fruits;
```

```
</script>
```

Output -

one,two,four,five

two,four,five

Q8. Mention iteration methods that loop through every elements in an array along with their return value.

–The `forEach()` method calls a function (a callback function) once for each array element.

```
<script>
```

```
const n = [5, 4, 3, 2];
```

```
let t = "";
```

```
n.forEach(myFunction);
```

```
document.getElementById("demo").innerHTML = t;
```

```
function myFunction(value, index, array) {
```

```
  t+= value + "<br>";
```

```
}
```

```
</script>
```

Output -

5

4

3

2

–The `map()` method creates a new array by performing a function on each array element.

```
<script>
```

```
const n = [5, 4, 3, 2];
```

```
const n2 = n.map(myFunction);
```

```
document.getElementById("demo1").innerHTML = n2;
```

```
function myFunction(value, index, array) {
```

```
  return value ;
```

```
}
```

```
</script>
```

Output -

5,4,3,2

–The reduce() method applies a function against an accumulator and each element in the array (from left to right) to reduce it to a single value

```
<script>
```

```
const n =[5, 4, 3, 2];
```

```
let sum = n.reduce(myFunction);
```

```
document.getElementById("demo").innerHTML = "The sum is " + sum;
```

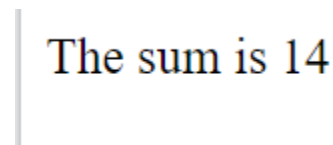
```
function myFunction(total, value, index, array) {
```

```
  return total + value;
```

```
}
```

```
</script>
```

Output -



–The filter() method creates a new array with all elements that pass the test implemented by the provided function.

```
<script>
```

```
const n = [5, 4, 3, 2];
```

```
const o3 = n.filter(myFunction);
```

```
document.getElementById("demo").innerHTML = o3;
```

```
function myFunction(value, index, array) {
```

```
  return value > 3;
```

```
}
```

```
</script>
```

Output -



Q9. Mention methods that return index(es) of array element(s).

–The indexOf() method searches an array for an element value and returns its position.

```
<script>
```

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
let index = fruits.indexOf("Apple");
```

```
document.getElementById("demo").innerHTML = index;
```

```
</script>
```

Output -

2

–lastIndexOf() returns the position of the last occurrence of the specified element.

```
<script>
const fruits = ["one", "five", "four", "five"];
let index = fruits.lastIndexOf("five");

document.getElementById("demo").innerHTML = index;
</script>
Output -
```

3

–The find() method returns the value of the first array element that passes a test function.

```
<script>
const n = [5, 4, 3, 2];

document.getElementById("demo").innerHTML = " number greater then 4 is " +
n.find(myFunction);

function myFunction(value, index, array) {
  return value > 4;
}
</script>
Output -
```

number greater then 4 is 5

–The findIndex() method returns the index of the first array element that passes a test function.

```
<script>
const n = [5, 4, 3, 2];

document.getElementById("demo").innerHTML = "Index of number greater then 4 is " +
n.findIndex(myFunction);

function myFunction(value, index, array) {
```

```
    return value > 4;
}
</script>
Output -
```



Index of number greater then 4 is 0

Q10. Mention methods used to check if element is present in array along with their return value.
–The some() method checks if some array values pass a test.

```
<script>
const n = [5, 4, 3, 2];
let someOver2 = n.some(myFunction);
```

```
document.getElementById("demo").innerHTML = "sum is more then 2 is " + someOver2;
```

```
function myFunction(value, index, array) {
    return value > 2;
}
</script>
Output -
```



sum is more then 2 is true

–The indexOf() method searches an array for an element value and returns its position.

```
<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let index = fruits.indexOf("Apple");
```

```
document.getElementById("demo").innerHTML = index;
```

```
</script>
Output -
```



2

–The find() method returns the value of the first array element that passes a test function.

```
<script>
const n = [5, 4, 3, 2];
```

```
document.getElementById("demo").innerHTML = " number greater then 4 is " +
n.find(myFunction);
```

```
function myFunction(value, index, array) {  
  return value > 4;  
}
```

</script>

Output -

number greater then 4 is 5

–The findIndex() method returns the index of the first array element that passes a test function.

<script>

```
const n = [5, 4, 3, 2];
```

```
document.getElementById("demo").innerHTML = "Index of number greater then 4 is " +  
n.findIndex(myFunction);
```

```
function myFunction(value, index, array) {  
  return value > 4;  
}
```

</script>

Output -

Index of number greater then 4 is 0

Q11. Mention methods used to access last element, part of array along with their return value.

Accessing the Last Element:

array[array.length - 1]: You can directly access the last element of an array using its length minus one as the index.

```
const a = [10, 20, 30, 40, 50];  
const lastElement = a[a.length - 1];  
console.log(lastElement);
```

Output -

50

array.at(): The at() method allows you to access an element at a specific index in the array.

Slicing a Part of an Array:

```
const arr1 = [1, 2, 3, 4, 5];  
console.log(arr1.at(2));
```

Output -

3

slice(): The slice() method returns a shallow copy of a portion of an array into a new array. It takes two arguments: the starting index (inclusive) and the ending index (exclusive).

```
<script>
```

```
const fruits = ["one", "two", "four", "five"];
```

```
const cut = fruits.slice(1);
```

```
document.getElementById("demo").innerHTML = fruits + "<br><br>" + cut;
```

```
</script>
```

Output -

one,two,four,five

two,four,five

Q12. Mention methods used to combine arrays.

The concat() method is used to combine two or more arrays, creating a new array that contains the elements of the original arrays in the order they appear. It does not modify the original arrays.

```
<script>
```

```
const mynumbers = ["one", "two", "four", "five"];
```

```
const myns = [5, 4, 3, 2];
```

```
const mytogether = myns.concat(mynumbers);
```

```
document.getElementById("demo").innerHTML = mytogether;
```

```
</script>
```

Output -

5,4,3,2,one,two,four,five

Q13. Sort a number array, array of objects based on string (eg: [{name: 'Zayn', age: 23 }, {name: 'Ana', age: 35 },...]) and reverse it.

```
<script>
  // Declaring the array objects
  var details = [
    { "name": "kiara", "age": 23 },
    { "name": "Alia", "age": 35 },
    { "name": "Sheena", "age": 30 },
```



```

];
//Sorting based on name
var sorteddetail = details.sort(function(a, b) {
    if (a.name > b.name) {
        return 1; // Compare strings for ascending order
    }
    if (a.name < b.name) {
        return -1;
    }
    return 0;
});

console.log("Sorted details:", sorteddetail);

// Reversing the sorted details
console.log(sorteddetail.reverse());

</script>

```

Output -

```

Sorted details:
▼ Array(3) ⓘ
  ► 0: {name: 'kiara', age: 23}
  ► 1: {name: 'Sheena', age: 30}
  ► 2: {name: 'Alia', age: 35}
  length: 3
  ► [[Prototype]]: Array(0)
▼ Array(3) ⓘ
  ► 0: {name: 'kiara', age: 23}
  ► 1: {name: 'Sheena', age: 30}
  ► 2: {name: 'Alia', age: 35}
  length: 3
  ► [[Prototype]]: Array(0)

```

Q14. In array: [1, 2, [3, 4], [5, [6, 7]]], mention methods used to get following output: [1, 2, 3, 4, 5, 6, 7].

The flat() method creates a new array with sub-array elements concatenated to a specified depth.

<p id="d9"></p>

<script>

```
const array = [1, 2, [3, 4], [5, [6, 7]]];
```

```
const newArr = array.flat();
```

```
document.getElementById("d9").innerHTML = newArr;
```

</script>

Output -

1,2,3,4,5,6,7

Q15. In array: [1, 2, 3, 4, 5], mention method to replace the last element with the first element.
– array.length - 1 refers to the index of the last element in the array. By assigning array[0] to this index, you're effectively replacing the last element with the value of the first element.

```
const array = [1, 2, 3, 4, 5];  
    array[array.length - 1] = array[0];  
  
    console.log(array);
```

▼ Array(5) ⓘ

0: 1

1: 2

2: 3

3: 4

4: 1

length: 5

Q16. In array: ["It's a sunny day", "I want ice-cream"] using appropriate method get this output:
["it's", "a", "sunny", "day", "I", "want", "ice-cream"] (Hint: "A boy".split(" ") => ["a", "boy"])

join(" ") method is used to join the elements of the array into a single string separated by spaces. The split(" ") method is used to split the string into an array of words based on the space character.

```
const array10 = ["It's a sunny day", "I want ice-cream"];  
    const array11 = array10.join(" ");  
    const outputArray = array11.split(" ");
```

```
console.log(outputArray);
```

Output -

```
▼ Array(7) ⓘ  
  0: "It's"  
  1: "a"  
  2: "sunny"  
  3: "day"  
  4: "I"  
  5: "want"  
  6: "ice-cream"  
  length: 7
```

Q.17 Use appropriate methods to convert string to array and vice versa
toString() converts an array to a string of separated by commas.

```
const array12 = ["welcome", "to", "trellissoft"];  
const string=array12.toString();  
console.log(string);
```

Output -

```
welcome,to,trellissoft p.html:48
```

The split() method splits a string into an array of substrings

```
const str = "welcome to trellissoft";  
const arr = str.split(" , ");  
console.log(arr);
```

Output -

```
▶ (3) ['welcome', 'to', 'trellissoft']      p.html:49
```