Q1. Head element of html usually contains meta information about the website , which is not visible to the user . this element contains following tags :

1. <title> - this tag is used to give name to web page .
2. <link> - now this is used to link any external files to ur current html page , this includes external css ,javascript , php .
3. <style> - this tag is used for internal css .
4. <script> - this tag is used for internal js [script tag can be used inside both head & body element]
5. <meta> - this tag gives metadata of the page like viewport[which helps in responsiveness] , characterset [UTF-8]
6. <no script> - this tag is used to display statement when any browser doesn’t support js
7. <base> - Honestly I didn't understand this tag . but i think its used for url of webpage

Q2. There are 3 ways in which we can add css in html page

1. External css - in this we create a separate css file [style.css] and using link tag we include css file in html
2. Internal css - this means using <style> tag in head element we can use css
3. Inline css - in this we write /use css directly in html tag [i.e <p style=”color: red”>Hello!! This is Shriya</p>]
   Highest Priority is given to inline css , then comes internal css , last is external css

Q3.Simpleselector - its basic selector which selects html elements based on their name , id , class

Eg :   Name selector —-   <p>Hi this is just an example </p>
                                                p{ color : red ;}

Id selector   —   <p id=”para1”>Hi this is just an example </p>
                                   #para1 {color: blue; }

Class selector   —-   <p class=”para2”>Hi this is just an example </p>
                                          .para2{text-align: left; }

Combinator selector - it selects tags/elements based on relation . now there are 4 types of relationship here

1. descendant selector - In this the selector selects all elements that are descendants of an specific html element Eg; if there are many <p> tags in a div that we can use this type of selector to give a specific style

   <div>
   <h1>hello</h1>

<p>Just trying this example</p>

<p>Just trying this example</p>

<p>Just trying this example</p>

<p>Just trying this example</p>

        </div>

css :   div p { text-align :right;}

2. Child selector - this selector selects all elements that are the children of a specific element

   css -   div > p{ font-style :cursive; }

3. adjacent sibling selector - this is used to select an element that is directly after another specific element.

   eg: <div>
         </div>
   <img>

   css -   div + img { float :right; }

4. general sibling selector - this selector selects all elements that are next siblings of a specified element.

   css-   div ~ p { background-color: yellow;}

Pseudo-classes selector : its used to define a special state of an element , its used for links styling ,table and so on . For links u can use a:hover, a:visited ,a:unvisited like styling

pseudo-element selector :its used to style specific parts of an element like the first letter , first line and so on. Its syntax is selector::pseudo-element {property: value;}

Q4. creating an array —-- let myArray = [3,7,9,12]; now for creating u can use var , let
Accessing array elements —- u can access any item in array by its index value .in array index value starts from 0 so in the above case 3 will have index value as 0 to access that we can simply write this statement — let item = myArray[0];
Manipulating an array – 1.add items in array — myArray.push(7);
2. adding items in beginning of array —-- myArray.unshift(4)
3. remove/deleting element —- myArray.pop(); , myArray.shift()

myArray.splice(2,1) // at index 2 remove 1 item , wth this method u can add or remove elements

recognizing if a variable is an array u can use isArray method

Q5. size of array : let myArray = [1,2,3,4,5];
console.log(myArray.length);

access First and last element of array : let myArray =myArray[0]; & for last element if u know the index value then u write directly or else u can use index as -1

adding new element : myArray.push (6);
myArray.unshift(12);   //adding 12 at beginning of array
myArray.splice(2,1,33)   // at index 2 adding 33
there is one more method concat but i didn't understand that

Q6 . adding new element : myArray.push (6);
myArray.unshift(12);   //adding 12 at beginning of array
myArray.splice(2,1,33)   // at index 2 adding 33
there is one more method concat but i didn't understand that

Q7.
There are many ways to delete an item from array

1. Using the Splice() method : in this u can delete elements from an array by specifying the index like where deletion should start and how many elements to remove

   let myArray = [1,2,3,4,5];
   myArray.splice(3,2);   // removing 2 elements at index 3 so o/p will be 1,2,3

2. Using the pop() method : this method removes the last element from an array
   myArray.pop();

3. using shift() method : myArray.shift() // this can delete the first element
4. using filter() method : this method creates a new array containing all elements that pass a test   …. i tried this method but didn't get it properly

```
   eg : let myArray = [1,2,3,4,5];
          myArray = myArray.filter(function(element)
{       return element !== 3;          // this will remove item which is equal to 3
}
);
```

Q8.

1. forEach() - this method calls a provided function once for each element in the array , in order , it doesn’t return a new array , its used for modifying the original array . it returns undefined

```
                    let myArray =[3,6,8,12,34];
                    myArray.forEach(function(element){
                    console.log(element);  } )
```

2. map() - this method will return a new array with the result of the function

   using same eg just changing the function

```
let myArray =myArray.map(function(element) {
return element *2; });
```

3. filter() - even this method return a new array

```
let myArray = [1,2,3,4,5];
   myArray = myArray.filter(function(element)
{       return element !== 3;          // this will remove item which is equal to 3}
```

4. every() - this meethod tests whether all elements in the array pass a test provided by a function. it returns ‘true’ if all elements pass the test

```
 let allGreaterThanZero = myArray.every(funtion(element){
return element > 0;  })
console.log(allGreaterThanZero) ;          //this will give true as ans
```

Q9. 1. indexof() - This method returns the index of the first occurrence of a specified element in the array. If the element is not found, it returns -1.

```
let myArray =[1,2,4,5];
```

2.lastIndexOf()

3. findIndex() - This method returns the index of the first element in the array that satisfies a provided testing function. If no element satisfies the condition, it returns -1.

same example as above   only function will change considering there is p tag with id para1

```
let index = myArray.findIndex(function(element){
```

```
return element > 2; })
document.getElementByID(“para1”).innerHtml= index;
```

Q10. 1. include() - The includes() method checks if an array includes a specific element. It returns true if the element is found and false otherwise.

```
var myArray = [1, 2, 3, 4, 5];
var elementToCheck = 3;

var isPresent = myArray.includes(elementToCheck);

console.log("Is " + elementToCheck + " present in the array? " + isPresent); // true
```

2. indexof() - The indexOf() method returns the index of the first occurrence of a specified element in the array, or -1 if the element is not found. You can use it to check for existence by comparing the result with -1.

```
var myArray = [1, 2, 3, 4, 5];
var elementToCheck = 6;

var index = myArray.indexOf(elementToCheck);

if (index !== -1) {
   console.log(elementToCheck + " is present at index " + index);
} else {
   console.log(elementToCheck + " is not present in the array.");
}
```

3. some()- method tests whether at least one element in the array passes a provided test in the form of a function. It returns true if at least one element satisfies the condition, and false if none do.

```
var myArray = [1, 2, 3, 4, 5];
var condition = function(element) {
   return element > 3;
};

var isPresent = myArray.some(condition);

console.log("Is there an element greater than 3 in the array? " + isPresent); // true
```

4. find() - The find() method returns the first element in the array that satisfies a provided testing function. It returns undefined if no element satisfies the condition.

```
var myArray = [1, 2, 3, 4, 5];
var condition = function(element) {
  return element === 3;
};

var foundElement = myArray.find(condition);

console.log("Is 3 present in the array? " + (foundElement !== undefined)); // true
```

Q11. Accessing the Last Element:

1. Using Indexing : U can access last element of an array by this method

   ```
   var myArray = [1, 2, 3, 4, 5];
   var lastElement = myArray[myArray.length - 1];
   console.log("Last element: " + lastElement);
   ```

1. Using pop() Method:
   The pop() method removes and returns the last element from an array. So, if you want to access the last element and remove it from the array, you can use pop().

   ```
   var myArray = [1, 2, 3, 4, 5];
   var lastElement = myArray.pop();
   console.log("Last element (and removed): " + lastElement);
   console.log("Array after popping: " + myArray);
   ```

   For just a portion of array

   1. Using Slice() method : this method extracts a section of an array n returns new array without modifying the original array .

   ```
   var myArray = [1, 2, 3, 4,]
   var portion =myArray.slice(1,4);
           console.log(portion)
   ```

   2.Using splice() - this can also be used to extract a portion , but this will modify the original array

   ```
   var myArray = [1, 2, 3, 4, 5];
   ```

```
var portion = myArray.splice(1, 3);
console.log("Portion of the array (and removed): " + portion);
console.log("Array after splicing: " + myArray);
```

Q12.

1. Array.from(): You can use the Array.from() method to create a new array from an iterable, such as an array-like object or another array. This method is particularly useful for combining multiple arrays.
   ```
   var array1 = [1, 2, 3];
   var array2 = [4, 5, 6];

   var combinedArray = Array.from(array1).concat(array2);

   console.log(combinedArray); // [1, 2, 3, 4, 5, 6]
   ```

2. concat(): The concat() method combines two or more arrays and returns a new array without modifying the original arrays.
   ```
   var array1 = [1, 2, 3];
   var array2 = [4, 5, 6];

   var combinedArray = array1.concat(array2);

   console.log(combinedArray); // [1, 2, 3, 4, 5, 6]
   ```

Q13. Sorting and Reversing a Number Array:

```
var numArray = [5, 2, 8, 1, 9];

// Sorting in ascending order
numArray.sort(function(a, b) {
  return a - b;
});

// Reversing the sorted array
numArray.reverse();

console.log(numArray); // [9, 8, 5, 2, 1]
```

Reversing an Array of Objects by a String Property:

```
var objArray = [
  { name: 'Zayn', age: 23 },
  { name: 'Ana', age: 35 },
```

```
    { name: 'John', age: 28 }
];

// Reversing the sorted array
objArray.reverse();

console.log(objArray);
```

Q14.

1. Using the flat() Method In modern JavaScript, you can use the flat() method to flatten an array. You can specify the depth to which the array should be flattened.

   ```
   var myArray = [1, 2, [3, 4], [5, [6, 7]]];

   var newArray = myArray.flat(Infinity);

   console.log(newArray); // [1, 2, 3, 4, 5, 6, 7]
   ```

Q15.

```
        var myArray = [1, 2, 3, 4, 5];

        // Replace the last element (5) with the first element (1)
        myArray[myArray.length - 1] = myArray[0];

        console.log(myArray); // [1, 2, 3, 4, 1]
```

Q16. var inputArray = ["It's a sunny day", "I want ice-cream"];

```
var myArray1 = inputArray.join()
var myArray2 = inputArray.split()

console.log(myArray2); // ["it's", "a", "sunny", "day", "i", "want", "ice-cream"]
```

Q17. Converting a String to an Array:

1. Using split(): You can split a string into an array using the split() method. This method takes a delimiter as an argument and splits the string into an array of substrings based on the delimiter.

```
var str = "Hello, world!";
var arr = str.split(', '); // Splitting by comma and space

console.log(arr); // ["Hello", "world!"]
```

2. Using Array.from(): You can convert a string to an array of individual characters using the Array.from() method.

```
var str = "Hello";
var arr = Array.from(str);

console.log(arr); // ["H", "e", "l", "l", "o"]
```

Converting an Array to a String:

1. Using join(): You can join the elements of an array into a string using the join() method. This method concatenates the array elements with a specified separator.
   ```
   var arr = ["Hello", "world!"];
   var str = arr.join(', '); // Joining with comma and space

   console.log(str); // "Hello, world!"
   ```

2. Using toString(): You can use the toString() method to convert an array to a string. This method joins the array elements using a comma as the separator.

```
var arr = ["Hello", "world!"];
var str = arr.toString();

console.log(str); // "Hello,world!"
```