ARC Tutorials Pvt Ltd
YouTube Channel: https://youtube.com/c/arctutorials
Email: Soorya.aaradhya@gmail.com

100+ JavaScript Code Snippets Every Developer Must Know

Solid collection of 100+ JavaScript code snippets that you must know and are very important for cracking any interview.

The book covers questions on Classes, Arrays, Strings, Sorting Algorithms, Objects, Promises and much more.

Become a better developer by solving logical code challenges.

The book will help you learn the most important aspects with code examples.

ABOUT THE BOOK

Our Mission

The mission of this book is to help you master all the frequently used JavaScript snippets every day or for cracking any interview.

The questions form the basic foundations for any logical work developers perform in day-to-day job roles.

The questions cover most aspects of JavaScript like Classes, Arrays, Strings, Promises, Objects, Sorting, Logical questions, Search Algorithms and much more.

About the team

This eBook is brought to you by the team at ARC Tutorials. We have spent more than 80 hours of effort in writing, compiling, and testing all the code snippets carefully.

You can reach the team at soorya.aaradhya@gmail.com

Make the most of the Book

To make the most of the book. Remember to practice often!

All the questions have been carefully selected, tested, and compiled for you. The best way to master it is by practicing.

Open any JavaScript editor, like https://jsfiddle.net/ and first try by yourself.

Then try the answers given in the document

Disclaimer

This book is the result of a lot of hard work and efforts put in compiling the top JavaScript coding interview questions asked in interviews.

The questions compiled from the team's knowledge as well as many various sources available on the internet.

Questions

Question #1:

How to define a class with properties and methods in JavaScript?

Answer

A class declared with a name which acts as its identifier. We can use the name to create new objects using the keyword new

```
class Car {
 constructor(model, name) {
   this.model = model;
   this.name = name;
 }
 start() {
   console.log(name + " engine started");
 }
}

car1 = new Car();
car1.start();
```

Question #2:

How to implement class inheritance in JavaScript?

Answer

A class can be extended by another class, and the new class will inherit all of its parent class methods. The object that initializes the child class will then be able to use methods from both classes.

```
class Car {
```

```
  constructor(model, name) {
    this.model = model;
    this.name = name;
  }
  start() {
    return "From Parent Start Function"
  }
}

class ElectricCar extends Car {
  start() {
          const fromParent = super.start(); // calling parent class constructor
    console.log(fromParent + " Battery at 95%.");
  }
}

let tesla = new ElectricCar("Tesla", "Model 3");
tesla.start();
```

Question #3:

How to find duplicate elements in a given array?

Given Input Array	[10, 5, 5, 10, 2, 3, 18]
Expected output	[5, 10]

Answer

To find duplicates in an array, we can make use of the Array filter method. Filter method takes 3 parameters, element, index and array on which filter is applied.

Then we check for the indexOf each element and return whichever does not match with the index.

```
const Ids = [10, 5, 5, 10, 2, 3, 18];

const duplicateIds = Ids.filter((e, i, a) => a.indexOf(e) !== i );
```

```
console.log(duplicateIds);
```

Question #4:

How to find the count of duplicates in an array?

Given Input Array	[‘Banana’, ‘April’, ‘May’, ‘Mark’, ‘May’, ‘Apple’, ‘May’]

Expected output	{
	App: 1,
	April: 1,
	Mark: 1,
	May: 3
	}

Answer

This one can be really tricky and can be asked in multiple variations. The fix will be the same for both number array as well as array of strings.

We will make use of the Array reduce method and using an object instance we will check if the key value is present in object if yes, increment or else return 1

```
const months = ["May", "April", "May", "Mark", "May", "App"];

const countOfMonths = months.reduce((obj, month) => {
  if(obj[month] == undefined){
    obj[month] = 1;
    return obj;
  }
  else {
    obj[month]++;
    return obj;
  }
```

```
}, {});

console.log(countOfMonths);
```

Question #5:

How to check if a given number is an integer?

Given Input Array	10.5
Expected output	true

Answer

To check if a given variable is integer, we can make use of the isNaN() method. This method can also be used to check if a variable is not an integer

```
const tax = 10.5; // true value
const str2 = "this is false"; // false value

console.log(!isNaN(tax));
console.log(!isNaN(str2));
```

Question #6:

Explain the difference between Object.freeze() vs const?

Answer

const applies to bindings ("variables"). It creates an immutable binding, i.e. you cannot assign a new value to the binding.

Object.freeze works on values, and more specifically, object values. It makes an object immutable, i.e. you cannot change its properties.

```
const person = {
   name: "Leonardo"
};

person = 'some';
console.log(person); // Uncaught TypeError: Assignment to constant variable.

let person = {
   name: "Leonardo"
};

Object.freeze(person);
person.name = "ARC"; // Uncaught SyntaxError: Identifier 'person' has already
been declared
```


Question #7:

How to Sort a Number Array?

Given Input Array	[10, 5, 12, 8, 30, 2, 20, 101]
Expected output	[2, 5, 8, 10, 12, 20, 30, 101]

Answer

To sort an array of numbers we will have to use a comparison method inside the sort method.

```
const studentIds = [10, 5, 12, 8, 30, 2, 20, 101];

studentIds.sort((a,b) =>{
   return a-b;
```

```
});

console.log(studentIds);
```

Question #8:

Sort a given array of strings

Given Input Array	['Mark','Ram', 'Larry', 'Adam', 'Sita', 'Lisa']
Expected output	["Adam", "Larry", "Lisa", "Mark", "Ram", "Sita"]

Answer

To sort an array of numbers we will have to use a comparison method inside the **sort** method.


```
const students = ['Mark','Ram', 'Larry', 'Adam', 'Sita', 'Lisa'];

console.log(students.sort());
```

Question #9:

How to find unique values in an array?

Given Input Array	[10, 5, 5, 10, 2, 3, 18]
Expected output	[10, 5, 2, 3, 18]

Answer

To find unique values in an Array we are going to use the Array **filter** method and check if the value is present using **indexOf** method.

```
const Ids = [10, 5, 5, 10, 2, 3, 18];

const uniqueIds = Ids.filter((e, i, a) => a.indexOf(e) == i );

console.log(uniqueIds);
```

Question #10:

How to find unique values from an Array in sorted order?

Given Input Array	[10, 5, 12, 8, 30, 2, 20, 101]
Expected output	[2, 5, 8, 10, 12, 20, 30, 101]

Answer

We will be using the Array **filter** method first and check if the element is present using the **indexOf** and then we will use the sort method to sort the Array elements.

```
const Ids = [10, 5, 5, 10, 2, 3, 18];

const uniqueIds = Ids.filter((e, i, a) => a.indexOf(e) == i ).sort((a,b) =>{
   return a-b;
});

console.log(uniqueIds);
```

Question #11:

Find maximum value in a numbered array?

Given Input Array	[10, 5, 12, 8, 30, 2, 20, 101]
Expected output	101

Answer

To find maximum value in a numbered array, we are implementing a method which will find Max value. Inside the method, we are making use of the Array **reduce** method and for each element, we will check if the value is greater than the previous one.

```
const studentIds = [10, 5, 12, 8, 30, 2, 20, 101];

function arrayMax(arr) {
  return arr.reduce(function (p, v) {
    return ( p > v ? p : v );
  });
}

console.log(arrayMax(studentIds));
```

Question #12:

Find minimum value in a numbered array?

Given Input Array	[10, 5, 12, 8, 30, 2, 20, 101]
Expected output	2

Answer

To find minimum value in a numbered array, we are implementing a method which will find minimum value. Inside the method, we are making use of the

Array **reduce** method and for each element, we will check if the value is less than the previous one.

```
const studentIds = [10, 5, 12, 8, 30, 2, 20, 101];

function arrayMin(arr) {
  return arr.reduce(function (p, v) {
    return ( p < v ? p : v );
  });
}

console.log(arrayMin(studentIds));
```

Question #13:

Find the average of the numbers in the numbered array?

Given Input Array	[10, 5, 12, 8, 30, 2, 20, 101]
Expected output	23.5

Answer

Using the Array reduce method, we will get the total count first, and then we will get the total length of the Array. Divide the total sum with the length of the array and we get the average of the numbers in the Array.

```
const studentIds = [10, 5, 12, 8, 30, 2, 20, 101];

const arrTotal = studentIds.reduce((a,b) => a + b, 0);

const arrLength = studentIds.length;

console.log(arrTotal/arrLength);
```

Question #14:

How can you uppercase the first character in a string array?

Answer

To capitalize the first character of every element in the Array, we will have to get the first character using **charAt** method and then apply **toUpperCase** and finally we will concatenate all the other characters using **substr**(1)

```
 const days = ['sunday', 'monday', 'tuesday', 'wednesday', 'thursday', 'friday',
'saturday'];

for (let day of days) {
  day = day.charAt(0).toUpperCase() + day.substr(1);
  console.log(day);
}
```


Question #15:

How to make a sentence out of the given string array?

Answer

We need to use the string **join**() method applied on the Array

```
var a = ['arc', 'tutorials'];
var sentence = a.join(" ");
console.log(sentence)
```

Question #16:

How to check if an array contains any element of another array?

Answer

To check if the array contains elements in another array, we will make use of the Array **includes** method to see if it includes any values from the Array.

```
let arr1 = [1, 2, 3];
let arr2 = [2, 3];

let isFounded = arr1.some( ai => arr2.includes(ai) );

console.log(isFounded);
```

Question #17:

Given two strings, how can you check if the strings are anagram for each other?

Answer

To sort an array of numbers we will have to use a comparison method inside the sort method.

```
var firstWord = "Mary";
var secondWord = "Army";

function isAnagram(first, second) {

  var a = first.toLowerCase();
  var b = second.toLowerCase();

  // Sort the strings, and join the resulting array to a string. Compare the results
  a = a.split("").sort().join("");
  b = b.split("").sort().join("");

  return a === b;
}
```

```
console.log(isAnagram(firstWord, secondWord));
```

Question #18:

How can you extract a few fields from the given JSON object and form a new array?

Given Input	const input = {
	"students": [{
	"studentName": "ARC Tutorials",
	"uuid": "124748ba-6fc4f"
	}, {
	"studentName": "Mark",
	"uuid": "1249b9ba-64d"
	}, {
	"studentName": "Lisa",
	"uuid": "124c78da-64"
	}, {
	"studentName": "Ram",
	"uuid": "124ee9da-6"
	}]
	};

Expected output	["ARC Tutorials", "Mark", "Lisa", "Ram"]

Answer

We will need to use the **map**() method, we will pass the element and then return only the required key from that element.

```
 const input = {
"students": [{
  "studentName": "ARC Tutorials",
  "uuid": "124748ba-6fc4f"
}, {
```

```
    "studentName": "Mark",
    "uuid": "1249b9ba-64d"
  }, {
    "studentName": "Lisa",
    "uuid": "124c78da-64"
  }, {
    "studentName": "Ram",
    "uuid": "124ee9da-6"
  }]
};

var op = input.students.map(function(item) {
  return item.studentName;
});

console.log(op);
```

Question #19:

Filter the given object based on certain conditions and return the corresponding object?

Given Input Object	**"Select the students whose rank is more than 10"**
	
	const input = {
	"students": [{
	"studentName": "ARC Tutorials",
	"rank": 10,
	"uuid": "124748ba-6fc4f"
	}, {
	"studentName": "Mark",
	"rank": 15,
	"uuid": "1249b9ba-64d"
	}, {
	"studentName": "Lisa",
	"rank": 30,
	"uuid": "124c78da-64"
	}, {
	"studentName": "Ram",

	"rank": 20, "uuid": "124ee9da-6" }] };

Expected output	[{
	rank: 15,
	studentName: "Mark",
	uuid: "1249b9ba-64d"
	}, {
	rank: 30,
	studentName: "Lisa",
	uuid: "124c78da-64"
	}, {
	rank: 20,
	studentName: "Ram",
	uuid: "124ee9da-6"
	}]

Answer

We will make use of the Array **filter** method and using if-else conditions we will check for conditions and return the element.

```
 const input = {
"students": [{
  "studentName": "ARC Tutorials",
  "rank": 10,
  "uuid": "124748ba-6fc4f"
}, {
  "studentName": "Mark",
  "rank": 15,
  "uuid": "1249b9ba-64d"
}, {
  "studentName": "Lisa",
  "rank": 30,
  "uuid": "124c78da-64"
}, {
  "studentName": "Ram",
```

```
    "rank": 20,
    "uuid": "124ee9da-6"
  }]
};
var op = input.students.filter(function(item) {
  if(item.rank > 10)
        return item
});

console.log(op);
```

Question #20:

Given an array of strings, reverse each word in the sentence?

Answer

We will be using the string methods reverse, split and join to reverse each word in the sentence

```
var string = "Welcome to ARC Tutorials";

// Output becomes slairotuT CRA ot emocleW
var reverseEntireSentence = reverseBySeparator(string, "");

// Output becomes emocleW ot CRA slairotuT
var reverseEachWord = reverseBySeparator(reverseEntireSentence, " ");

function reverseBySeparator(string, separator) {
  return string.split(separator).reverse().join(separator);
}

console.log(reverseEntireSentence)
console.log(reverseEachWord)
```

Question #21:

How to check if an object is present in an Array or not?

Answer

Most modern browsers support Array method isArray, so we can use this method to determine if a given object is an Array or not.

```
const studentIds = [10, 5, 12, 8, 30, 2, 20, 101];

console.log(Array.isArray(studentIds));
```

Question #22:

How to empty an array?

Answer

To empty an Array, just assign an empty [] array to the variable.

```
var arr1 = ['a','b','c','d','e','f'];
arr1 = [];
console.log(arr1);
```

Question #23:

What is IIFEs (Immediately Invoked Function Expressions)?

Answer

It's an Immediately-Invoked Function Expression, or IIFE for short. It executes immediately after it's created.

```
(function IIFE(){
        console.log( "Hello!" );
})();
// "Hello!"
```

Question #24:

How do you sort and reverse an array without changing the original array?

Answer

We can use **slice**() to make a copy then **reverse**() it


```
var arr1 = ['a','b','c','d','e','f'];

var newarray = arr1.slice().reverse();

console.log(newarray);
```

Question #25:

Write a function to check if a given string is Palindrome or not?

Answer

We are going to write a custom method which includes usage of string methods like replace, toLowerCase, reverse and join

```
function isPalindrome(str) {
  str = str.replace(/\W/g, '').toLowerCase();
  return (str == str.split('').reverse().join(''));
}

console.log(isPalindrome('teet')); // true
console.log(isPalindrome('teer')); // true
```

Question #26:

How can you combine two Arrays into a third Array using spread operator?

Answer

To combine two or more Arrays, we will use the spread operator.

```
const first = [1,2]
const second = [3,4]
const third = [...first, ...second]
```

Question #27:

How can you Insert an element at a specific index in an Array?

Answer

To insert an element in an Array at a specific index, we be using both spread operator and Array **slice** method

```
const  original = [1,2,4];
const el = 3;
const insertAt = 2;
```

```
const newArray = [...original.slice(0,insertAt), el, ...original.slice(insertAt)]

console.log(newArray)
```

Question #28:

How can you Replace an element at a specific index in an Array?

Answer

To replace an element in an Array at a specific index, we be using both spread operator and Array **slice** method

```
const  original = [1,2,4,4]
const el = 3
const replaceAt = 2
const newArray = [...original.slice(0,replaceAt ), el, ...original.slice(replaceAt + 1)]

console.log(newArray);
```

Question #29:

How can you Delete an element at a specific index in an Array?

Answer

To delete an element in an Array at a specific index, we be using both spread operator and Array **slice** method

```
const  original = [1,2,'x',3]
```

```
const deleteAt = 2
const newArray = [...original.slice(0,deleteAt), ...original.slice(deleteAt + 1)]

console.log(newArray);
```

Question #30:

How can you delete a specific element?

Answer

To delete a specific element in an Array, we be using both spread operator and Array **filter** method

```
const original = [1,2, 5, 3]
const toDelete = 5
const newArray = original.filter(item => item !== toDelete)

console.log(newArray)
```

Question #31:

How do you clone an Object?

Answer

We can clone an Object using multiple ways including Object.assign method or with spread operator etc. Here we are showing you way to do with **Object.assign** method

```
const students = { studentId: '10', studentName: 'ARC Tutorials' };

const cloneStudents = Object.assign({}, students);
```

```
console.log(cloneStudents);
```

Question #32:

How do you add an element at the beginning of an array?

Answer

We can use the Array **unshift** method instead. We are going to show you another way of adding an element at the beginning of any array using the **spread operator**

```
const original = [1,2,3]

const el = 4

const newArray =  [el, ...original]

console.log(newArray);
```

Question #33:

How to remove an element from the end of the array?

Answer

For this problem, we can either use **spread operator** or Array **splice** method

```
const students = [1,2,3]

const newStudentArr = students.slice(1)

console.log(newStudentArr)
```

Question #34:

How can you split a string into an Array?

Answer

To solve this problem, we will need to use the string **split** method to split each word after an empty space

```
const msg = "Welcome to ARC Tutorials";

const arr = msg.split(' ');

console.log(arr);
```

Question #35:

How can you remove an element from the beginning of the array?

Answer

We will be using the Array **splice** method to remove an element from the beginning of the Array.

```
const original = [1,2,3]
const newArray = original.slice(0,-1)
```

Question #36:

How can you add an element to an object?

Answer

To add a new element in an Object, we make use of the spread operator

```
const original = {
  street_address: '123 Main Street',
  city: 'New York',
  state: 'NY',
  zip: '10005'
}

// add a new field
const withAdditionalInfo = {
   ...original,
  'street_address_2' : 'Apt 456',
}

console.log(withAdditionalInfo)
```

Question #37:

How can you replace an existing element in an object?

Answer

To replace an existing element in an Object, we make use of the spread operator and just override the element in the object with the new value

```
const original = {
  street_address: '123 Main Street',
  city: 'New York',
```

```
  state: 'NY',
  zip: '10005'
}

//replace an existing field
const withReplacedZip = {
   ...original,
   'zip' : '10005-1546',
}

console.log(withReplacedZip)
```

Question #38:

How can you combine two objects?

Answer

To combine two or more objects, we will need to use the spread operator.

```
const original = {
  street_address: '123 Main Street',
  city: 'New York',
  state: 'NY',
  zip: '10005'
}

//replace an existing field
const withReplacedZip = {
   ...original,
   'zip_updated_code' : '10005-1546',
}

console.log(withReplacedZip)
```

Question #39:

How do you write a function which can take (x) number of parameters?

Answer

ES6 introduced the REST operator, using which we can pass any number of arguments to methods.

```
function add(...args){
  console.log(...args);
}

add(3,4); // can take 2 arguments
add(5,6,7,8); // can take 4 arguments
```


Question #40:

Convert the given number into the exact decimal points to the right side?

Answer

To sort an array of numbers we will have to use a comparison method inside the sort method.

```
const num = 435.78787878;
const n = num.toFixed(2);

console.log(studentIds);
```

Question #41:

How do you return a character from a string at a specific index(2)?

Answer

If we want to return the character at index(2) - we will have to use string **charAt** method

```
let text = "Welcome to ARC Tutorials";
let letter = text.charAt(2);
console.log(letter);
```

Question #42:

How do you create an Array out of a given sentence?

Answer

To create an Array out of a given sentence, we will make use of the string **split** method.

```
var str = "Welcome to ARC Tutorials Javascript Interview Questions";
var words = str.split(" ");
console.log(words);
```

Question #43:

How do you replace a given string in the string of arrays?

Answer

To replace a given string in the string of Arrays, we will use map method to check for each string in the Array and then use replace to check if the string is present in the string.

In the below code snippet, we are trying to find the string “er” and replace it with “”

```
let array = ["erf,","erfeer,rf","erfer"];
array = array.map(function(x){ return x.replace(/er/g,"") });

console.log(array);
```

Question #44:

How do you write an add() function using javascript currying concept?

Answer

Currying is a technique of evaluating a function with multiple arguments, into a sequence of functions with single/multiple arguments.

```
function add(x){
   let sum = x;
   function resultFn(y){
      sum += y;
      return resultFn;
   }
   resultFn.valueOf = function(){
         return sum;
      };
   return resultFn;
}

console.log(add(2)(3).valueOf())
```

Question #45:

Implement a groupBy method in JavaScript?

Given Input Array	const movies = [
	  { title: "Sonic the Hedgehog", year: 2020 },
	  { title: "Mulan", year: 2020 },
	  { title: "Godzilla vs. Kong", year: 2021 },
	];

Expected output	const moviesByYear = {
	  2020: [
	  { title: "Sonic the Hedgehog", year: 2020 },
	  { title: "Mulan", year: 2020 },
	  ],
	  2021: [{ title: "Godzilla vs. Kong", year: 2021 }],
	};

Answer

To write a function that will group an array of objects by its key value, you need to use the array reduce() method to iterate over the array and put each array element into the right place.

```
function groupBy(arr, criteria) {
  const newObj = arr.reduce(function (acc, currentValue) {
    if (!acc[currentValue[criteria]]) {
      acc[currentValue[criteria]] = [];
    }
    acc[currentValue[criteria]].push(currentValue);
    return acc;
  }, {});
  return newObj;
}

const movies = [
  { title: "Sonic the Hedgehog", year: 2020 },
  { title: "Mulan", year: 2020 },
```

```
  { title: "Godzilla vs. Kong", year: 2021 },
];

const moviesByYear = groupBy(movies, "year");

console.log(moviesByYear);
```

Question #46:

Explain WeakSet in javascript with an example?

Answer

Weakset contains only objects and no other type.

```
let obj1 = {message:"Hello world"};
const newSet3 = new WeakSet([obj1]);
console.log(newSet3.has(obj1)); // true
```

Question #47:

Explain WeakMap in javascript with an example?

Answer

In a WeakMap, The keys and values in weakmap should always be an object.

```
let obj = {name:"Vivek"};
const map3 = new WeakMap();
```

```
map3.set(obj, {age:23});
```

Question #48:

Explain what is Object Destructuring with an example?

Answer

Object destructuring is a new way to extract elements from an object or an array.

```
const hero = {
  name: 'ARC Tutorials',
  realName: 'Soorya'
};
const { name, realName } = hero;

console.log(realName)
```

Question #49:

How can we generate a random alphanumeric string in JavaScript?

Answer

We can generate a random alphanumeric string with JavaScript by first creating a variable that contains all the characters you want to include in the random string

```
const characters =

"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
const length = 9;
let randomStr = "";

for (let i = 0; i < length; i++) {
  const randomNum = Math.floor(Math.random() * characters.length);
  randomStr += characters[randomNum];
}

console.log(randomStr); // will be different with each execution
```

Question #50:

How can we call a function which logs a message after every 5 seconds?

Answer

To call a function after 5 seconds, we can make use of the **setInterval** method. Using the **setInterval** method, we can specify the time delay and the function to be called after the time lapse.

```
function myFunc() {
   console.log("This function will be called with delay");
}

setInterval(function() { myFunc(); }, 5000);
```

Question #51:

How can we delay calling a function after 5 seconds?

Answer

We can use setTimeout to delay a function call by specifying the time in milliseconds and the method to call the time interval.

```
function myFunc() {
   console.log(“This function will be called with delay”);
}

setTimeout(function() { myFunc(); }, 5000);
```

Question #52:

Write a function that performs binary search on a sorted array?

Answer

Below we are implementing a function named binarySearch which will accept an array and target value as arguments.

```
 function binarySearch(array, targetValue) {
  var min = 0;
  var max = array.length - 1;
  var guess;

  while(min <= max) {
     guess = Math.floor((max + min) / 2);

     if (array[guess] === targetValue) {
        return guess;
     }
     else if (array[guess] < targetValue) {
        min = guess + 1;
     }
     else {
        max = guess - 1;
     }
```

```
    }

    return -1;
}

var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9];

console.log(binarySearch(arr, 6));
```

Question #53:

How can we parse a given JSON object?

Answer

To sort an array of numbers we will have to use a comparison method inside the sort method.


```
const json = '{"result":true, "count":42}';
const obj = JSON.parse(json);
console.log(obj);
```

Question #54:

How do you check whether a string contains a substring?

Answer

To find if a substring exist in a string, we can use string **includes** method

```
const string = "foo";
const substring = "oo";

console.log(string.includes(substring)); // true
```

Question #55:

How do I get query string values in javascript?

Answer

To get query params from the URL, we can use URLSearchParams and pass the url string.

```
const url= new URL('https://test.com?mode=night&page=2&time=day');


const urlSearchParams = new URLSearchParams(url.search);
const params = Object.fromEntries(urlSearchParams.entries());

console.log(params)
```


Question #56:

How to create and trigger events in javascript?

Answer

Events can be created with the Event constructor as follows:

```
var event = new Event('build');

// Listen for the event.
elem.addEventListener('build', function (e) { /* ... */ }, false);

// Dispatch the event.
elem.dispatchEvent(event);
```

Question #57:

How to display the current date in javascript?

Answer

In order to display the current date, first we need to create an instance of Date object and then using the methods **getDate**, **getMonth** and **getYear** extract the details required.

```
var today = new Date();
var dd = String(today.getDate()).padStart(2, '0');
var mm = String(today.getMonth() + 1).padStart(2, '0'); //January is 0!
var yyyy = today.getFullYear();

today = mm + '/' + dd + '/' + yyyy;
console.log(today);
```

Question #58:

How do you compare time for two dates?

Answer

We need to extract the time using the **getTime** method and compare both values.

```
const d1 = new Date('2013-05-23');
const d2 = new Date('2013-05-23');

var same = d1.getTime() === d2.getTime();

console.log(same);
```

Question #59:

How do you check if a string starts with another string?

Answer

To check if a string starts with given string, we can make use of the string **startsWith** method

```
let text = "Hello world, welcome to the universe.";
const doesValueStart = text.startsWith("Hello");

console.log(doesValueStart);
```

Question #60:

How do you remove whitespaces from a given string?

Answer

To remove whitespace from a given string, we can use the string replace method.

```
const str = "Welcome to ARC Tutorials";

const removeWhiteSpace = str.replace(/\s/g, "");

console.log(removeWhiteSpace);
```

Question #61:

How do you assign default values to variables?

Answer

We can set default values using either the logical operator or Ternary operator.

```
let isHappyHour = false;

// Logical Operator
isHappyHour = isHappyHour || true;
console.log(isHappyHour);

// Ternary
isHappyHour = isHappyHour ? isHappyHour : true;
console.log(isHappyHour);
```

Question #62:

For a given function, count the number of parameters expected by a function?

Answer

We can use length property with the name of the function and we will get the number of arguments the function can accept.

```
function test( a, b, c ){
   console.log(a);
   console.log(b);
   console.log(c);
}

console.log(test.length);
```

Question #63:

Implement a method which generates 5 random numbers?

Answer

We will be using the Math's floor and random methods to generate the random numbers.

```
const arr = [];

while(arr.length < 8){
    var r = Math.floor(Math.random() * 100) + 1;
    if(arr.indexOf(r) === -1) arr.push(r);
}

console.log(arr);
```

Question #64:

Implement a method which generates random numbers between 41 and 67 and sort them reverse?

Answer

To generate random numbers between a minimum and maximum values, we will use Math **floor** and **random** methods. Next we will have to use the **sort** method for sorting and finally reverse them using the **reverse** method.

```
var arr = [];
const min = 41;
const max = 67;
while(arr.length < 8){
   const rndInt = Math.floor(Math.random() * (max - min + 1) + min)
   arr.push(rndInt);
}
console.log(arr)
console.log(arr.sort())
console.log(arr.reverse());
```

Question #65:

How do you search a string for a pattern?

Answer

We can perform a string search using either match method or search method.

Using the **search** method, the result will be the index of the string matched

Using the **match** method, we will get the string matched.

Below are examples of the same.

```
// Approach #1  -> using search method

const text = "Welcome to ARC Tutorials Javascript snippet code";
const result = text.search(/tutorial/i);

console.log(result); // returns the index of matched string

// Approach #2 -> using match method with regex pattern

const text = "Welcome to ARC Tutorials Javascript snippet code";
const result = text.match(/tutorial/i);

console.log(result); // returns the matched string
```

Question #66:

How to write an Object and implement multiple function chaining?

Answer

We will need to make use of the "**this**" operator inside the object for reference.

We can write any number of methods inside the object and using this reference we can chain multiple functions.

```
var Obj = {
  result: 0,
  addNumber: function(a, b) {
    this.result = a + b;
    return this;
  },

  multiplyNumber: function(a) {
    this.result = this.result * a;
    return this;
  }
};

Obj.addNumber(10, 20).multiplyNumber(10)
console.log(Obj.result)
```


Question #67:

How do you determine whether an object is frozen or not?

Answer

To check if an object is freezed or not, we need to make use of the method Object.isFrozen and pass the object to check.

```
const Obj = {
  result: 0
};

//Object.freeze(Obj); // uncomment this line to set object to freeze

const result = Object.isFrozen(Obj);

console.log(result);
```

Question #68:

How do you determine two values are the same or not using an object?

Answer

To determine if two values are the same or not, we will use Object's is method which will check if 2 values are the same and return a boolean value.

```
Object.is('hello', 'hello');     // truef
Object.is(window, window);   // true
Object.is([], []) // false
```


Question #69:

How do you copy properties from one object to another object?

Answer

To copy properties from one object to another one, we will use the **Object.assign** method.

```
const target = {};
const sources = {
  userId: 10,
  userName: 'ARC Tutorials'
};
Object.assign(target, sources) //

console.log(target);
```

Question #70:

How do you determine if an object is sealed or not?

Answer

To determine if an object is sealed or not, we will use the **Object.isSealed** method.

```
const sources = {
  userId: 10,
  userName: 'ARC Tutorials'
};

Object.seal(sources);
console.log(Object.isSealed(sources))
```

Question #71:

How do you get an enumerable key and value pairs?

Answer

To get enumerable key and value pairs from an Object, we will use the **Object.entries** method.

```
const object = {
  a: 'Arc Tutorials',
  b: 100
};

for (let [key, value] of Object.entries(object)) {
```

```
  console.log(`${key}: ${value}`);
}
```

Question #72:

What is the main difference between Object.values and Object.entries method?

Answer

The main difference is that the **Object.values()** method returns only all the own values while **Object.entries()** method returns an array of arrays with key and value and it works from ES6

```
const object = {
   a: 'Good morning',
   b: 100
 };

 for (let value of Object.values(object)) {
   console.log(`${value}`); // 'Good morning'
 }


let valuesArray = Object.entries(object);

for (let value of valuesArray) {
  console.log(value)
}
```

Question #73:

How can you get the list of keys of any object?

Answer

To get the list of keys in an Object, we will use the **Object.keys** method

```
const user = {
  a: 'Arc Tutorials',
  b: 100
};

console.log(Object.keys(user));
```

Question #74:

How do you encode an URL?

Answer

We can easily encode URLs using the built-in **encodeURIComponent** method.

```
var myUrl = "http://example.com/index.html?param=1&anotherParam=2";
var myOtherUrl = "http://example.com/index.html?url=" +
encodeURIComponent(myUrl);

console.log(myOtherUrl);


    console.log(studentIds);
```

Question #75:

How do you define property on an Object constructor?

Answer

To define a new property on an Object, we can make use of **Object.defineProperty** method

```
const newObject = {};

Object.defineProperty(newObject, 'newProperty', {
  value: 100,
  writable: true
});

console.log(newObject.newProperty);
```

Question #76:

How can you call the constructor of a parent class?

Answer

To call the constructor of a parent class, in the child class, we need to call **super** method. Super method refers to the parent's constructor or can also call parent's functions.


```
class Rectangle {
  constructor(width, height){
  console.log("I am parent" + width + height)
  }
}

class Square extends Rectangle {
  constructor(length) {
    super(length, length);
    this.name = 'Square';
  }

  get area() {
    return this.width * this.height;
  }

  set area(value) {
    this.area = value;
  }
}
```

```
const sq = new Square(10);
```

Question #77:

How do you check whether an object can be extendable or not?

Answer

To check if an object is extendable or not, we will use the **Object.isExtensible** method.

```
const sources = {
   name: 'ARC Tutorials'
};

console.log(Object.isExtensible(sources)); //true
```

Question #78:

How do you prevent an object from extending?

Answer

To prevent an object from extending we will have to use **Object.preventExtensions** method

```
const sources = {
        name: 'ARC Tutorials'
};

Object.preventExtensions(sources);
```

```
console.log(Object.isExtensible(sources))
```

Question #79:

How do you find the Vowels?

Answer

To find vowels in a given string, we will need to implement a method. The method will check for the vowels passed through an Array and using the Array **includes** method, we can check if the vowels are present in the string.

```
function findVowels(str) {
  let count = 0
  const vowels = ['a', 'e', 'i', 'o', 'u']
  for(let char of str.toLowerCase()) {
    if(vowels.includes(char)) {
      count++
    }
  }
  return count
}

const result = findVowels("this is some string");
console.log(result);
```

Question #80:

What are default values in destructuring assignment?

Answer

Below is an example of how to assign default values in destructuring assignment.

```
const {x=2, y=4, z=6} = {x: 10};

console.log(x); // 10
console.log(y); // 4
console.log(z); // 6
```

Question #81:

How do you swap variables in destructuring assignment?

Answer

To swap variables, we can just use the desturing assignment in the opposite way.

```
var x = 10, y = 20;

[x, y] = [y, x];

console.log(x); // 20
console.log(y); // 10
```

Question #82:

How do you combine two or more arrays?

Answer

We can combine two or more Arrays using the Array concat method.

```
var veggies = ["Tomato", "Carrot", "Cabbage"];
var fruits = ["Apple", "Orange", "Pears"];

var veggiesAndFruits = veggies.concat(fruits);

console.log(veggiesAndFruits);
```

Question #83:

How to create a specific number of copies for a string?

Answer

To repeat a string, we can use the string **repeat** method.

```
const result = "ARC Tutorials".repeat(3);

console.log(result); // will repeat the string 3 times
```

Question #84:

What is the easiest way to convert an array to an object?

Answer

The easiest way to convert an Array into an Object is using the rest operator.

```
var fruits = ["banana", "apple", "orange", "watermelon"];
var fruitsObject = {...fruits};
console.log(fruitsObject)
```

Question #85:

Verify that a function argument is a Number or not?

Answer

To verify that a function argument is a number or not, we will need to pass an argument to the function. Using the **isNaN**, **parseFloat** and **isFinite** method we can determine if the value is number or not.

```
function isNumber(n){
    return !isNaN(parseFloat(n)) && isFinite(n);
}

console.log(isNumber('s'));
```

Question #86:

What is the easiest way to resize an Array?

Answer

The easiest way to resize an Array is by assigning a new value to the length property of the Array.

```
var array = [1, 2, 3, 4, 5];
console.log(array.length); // 5

array.length = 2;
console.log(array.length); // 2
console.log(array); // [1,2]
```

Question #87:

What's the difference between a function expression and function declaration?

Answer

The main difference between a function expression and a function declaration is the function name, which can be omitted in function expressions to create anonymous functions.

```
hoistedFunc();
notHoistedFunc();

function hoistedFunc(){
  console.log("I am hoisted");
}

var notHoistedFunc = function(){
  console.log("I will not be hoisted!");
}
```

Question #88:

How to detect a mobile device with JavaScript?

Answer

We will make use of the navigator object to detect the mobile devices

```
iOS: function() {
   return navigator.userAgent.match(/iPhone|iPad/i); // check for iPhone/iPad
}

Android: function() {
   return navigator.userAgent.match(/Android/i); // check for Android
}
```


Question #89:

How to fill static values in an array?

Answer

We can fill static values in an array using the fill() method in JavaScript.

```
var arr = ["Welcome", "to", "ARC", "Tutorials"];
console.log(`Initial Array: ` + arr);

console.log(`Updated array ` + arr.fill("ARC Tutorials Channel",1,2));
```

Question #90:

Why do we use The some() method in Arrays?

Answer

The some() method checks if an array element satisfies a particular given condition.

```
const val = [20, 28, 45];
function display(a) {
    return a > 21;
}
console.log("Are their values greater than 21 in the array = "+val.some(display));
```

Question #91:

How to add 15 minutes to a JavaScript Date?

Answer

To add “n” number of minutes to JavaScript date object, we will need to first get the date and then using the The getMinutes() method get the current minutes in JavaScript.

Finally using the setMinutes() method, we will set the new value.

```
var date = new Date();
console.log("Current Date and Time = " + date);

date.setMinutes(date.getMinutes() + 15 );
console.log("New Date = "+date);
```

Question #92:

Explain JavaScript Promises with an example?

Answer

Below is an example of a JavaScript promise.

```
const myPromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('Resolved');
  }, 300);
});

myPromise
.then((v) => console.log(v))
.catch(() => console.log("Error Handling"))
```

Question #93:

Which keyword can be used to deploy inheritance in ES6?

Answer

The extends keyword is used to implement inheritance in the ES6 language.

```
 class Classroom {
    constructor(students) {
        this.students = students;
    }
    room() {
        console.log('This class has  ' + this.students + ' students');
    }
}

class sectionA extends Classroom {
    constructor(students) {
        super(students);
    }
    sec() {
```

```
        console.log('section A');
    }
}

let secA = new sectionA(40);

secA.room();
secA.sec();
```

Question #94:

What is the difference between for..of and for..in?

Answer

for in: runs over an object's enumerable property names.

for of: (new in ES6) takes an object-specific iterator and loops through the data it generates.

Both the for..of and for..in commands iterate over lists, but the results they return are different: for..in returns a list of keys on the object being iterated, whereas for..of returns a list of values of the object's numeric attributes.

```
let arr = [3, 4, 5];

for (let i in arr) {
  console.log(i); // "0", "1", "2",
}

for (let i of arr) {
  console.log(i); // "3", "4", "5"
}
```

Question #95:

How to set a property in localStorage?

Answer

Below is the snippet to set a property in localStorage

```
window.localStorage.setItem("grade","One");
```

Question #96:

Give a list of the various ways using which an HTML element can be accessed within a JavaScript code?

Answer

Below are some of the methods using which we can access the HTML elements within JavaScript code.

> **getElementById**('idname'): Using this method, you can get an element by the ID name of the element.
>
> **getElementsByClass**('classname'): Using this method, you can get all elements which have a given classname.
>
> **getElementsByTagName**('tagname'): Using this method, you can get all elements which have a given tag name.
>
> **querySelector**(): The querySelector() function takes the css style selector and returns the first selected element

---

Question #97:

State the difference between Apply and Call?

Answer

The **call**() method helps to call a function which has a given ‘this’ value and the arguments which are individually provided

```
fun.call(thisArg[, arg1[, arg2[, …]]])
```

The apply() method is used to call a function which has a given ‘this’ value but the arguments are presented as an array

```
fun.apply(thisArg, [argsArray])
```

Question #98:

What is the difference between undefined and not defined in JavaScript?

Answer

In JavaScript, if you try to use a variable that doesn't exist and has not been declared, then JavaScript will throw an error var name is not defined

If a variable that is neither declared nor defined, when we try to reference such a variable we'd get the result not defined

```
var x; // declaring x
console.log(x); //output: undefined

console.log(y);  // Output: ReferenceError: y is not defined
```

Question #99:

Explain Closure in JavaScript with an example?

Answer

Closure means that an inner function always has access to the vars and parameters of its outer function, even after the outer function has returned.

In the below example, return InnerFunction; returns InnerFunction from OuterFunction when you call OuterFunction(). A variable innerFunc referenced the InnerFunction() only, not the OuterFunction(). So now, when you call innerFunc(), it can still access outerVariable which is declared in OuterFunction(). This is called Closure

```
function OuterFunction() {

    var outerVariable = 100;

    function InnerFunction() {
        console.log(outerVariable);
    }

    return InnerFunction;
}
var innerFunc = OuterFunction();

innerFunc(); // 100
```

Question #100:

How to calculate the length of an associative array using JavaScript?

Answer

We can calculate the length in different ways. In the below snippet, we are calculating the length using the Object's keys method.

```
const arr = {
        "apple": 10,
        "grapes": 20
    };

const length = Object.keys(arr).length;
console.log(length)
```

Question #101:

Explain JavaScript promise.all with an example?

Answer

Whenever we need to wait for all promises, that time we will make use of Promise.all(). The callback is executed only when all the states are successful.then()If one fails, execute it.catch().

```
var p1 = Promise.resolve(1)
var p2 = Promise.resolve(2)
var p3 = Promise.resolve(3)
let pro =  Promise.all([p1, p2, p3])
  .then(res => {
      console.log(res)  //  [1, 2, 3]
  })
  .catch( err => {
      console.log("Failed")
  })
console.log(pro)
```

Question #102:

Explain Promise.race with an example?

Answer

Promise.race()It is also used to receive a set of asynchronous tasks, and then execute them in parallel. Only the result of the fastest asynchronous operation is retained. Other methods are still executing, but the execution result will be discarded

```
var p1 = new Promise(function(resolve, reject) {
    setTimeout(reject, 500, "one");
});
var p2 = new Promise(function(resolve, reject) {
    setTimeout(resolve, 100, "two");
});

Promise.race([p1, p2])
.then(res => {
          console.log('then', res)
})
.catch( err => {

console.log('catch', err)   // `catch error`
 return err
})
```

Question #103:

Give an example of an Anonymous function?

Answer

As the name states, it is a function without a name. They are declared during runtime dynamically using a function operator since it offers more flexibility than a declarator.

```
var display=function()
```

```
{
 console.log("Anonymous Function is declared");
}
display();
```

Question #104:

What is Prototype Property? Explain with an Example.

Answer

Prototype property is usually used for implementing inheritance. Every function has one, by default the value of which is null. Methods and properties are added to the prototype to make it available to the instances.

```
function Rectangle(x, y) {

    this.x = x;

    this.y = y;

}

Rectangle.prototype.perimeter = function() {

   return 2 * (this.x + this.y);

}

var rectangle = new Rectangle(4, 3);

console.log(rectangle.perimeter()); // outputs '14'
```

Question #105:

Explain function hoisting with an example?

Answer

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution.

Hoisted functions can be used before their declaration. As you can see, the JavaScript interpreter allows you to use the function before the point at which it was declared in the source code. This is extremely useful as the function can be called before defining it. One can then define the function anywhere in the program code

```
Func_Hoisted(); // function called before definition

function Func_Hoisted() {
    console.log("This function is hoisted!");
}
```


Question #106:

How to implement Bubble sort in JavaScript?

Answer

The below given code snippet will perform Array sorting using the bubble sort algorithm.

```
function bblSort(arr){

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

   // Last i elements are already in place
   for(var j = 0; j < ( arr.length - i -1 ); j++){

     // Checking if the item at present iteration
```

```
    // is greater than the next iteration
    if(arr[j] > arr[j+1]){

      // If the condition is true then swap them
      var temp = arr[j]
      arr[j] = arr[j + 1]
      arr[j+1] = temp
    }
  }
 }
 // Print the sorted array
 console.log(arr);
}

// This is our unsorted array
var arr = [234, 43, 55, 63,  5, 6, 235, 547];

bblSort(arr);
```


Question #107:

How to find prime numbers in an Array?

Answer

The below code snippet will find all the prime numbers in a given Array.

```
var numArray = [2, 3, 4, 5, 6, 7, 8, 9, 10]

numArray = numArray.filter((number) => {
  for (var i = 2; i <= Math.sqrt(number); i++) {
    if (number % i === 0) return false;
  }
  return true;
});

console.log(numArray);
```

Question #108:

How to generate fibonacci series in JavaScript?

Answer

Below code snippet will generate fibonacci series in JavaScript.

```
var i;
var fib = []; // Initialize array!

fib[0] = 0;
fib[1] = 1;

for (i = 2; i <= 10; i++) {
  // Next fibonacci number = previous + one before previous

  fib[i] = fib[i - 2] + fib[i - 1];
  console.log(fib[i]);
}
```


RESOURCES

Contact Us

If you have any questions or have any suggestions or editorial correction. You can reach us in multiple ways.

- Email us at: soorya.aaradhya@gmail.com
- YouTube Channel: https://www.youtube.com/c/arctutorials