All front end Interview questions asked during my recent job hunt.

#javascript #react #career

Abhijeet Yadav

Aug 4 *Updated on Aug 20, 2020* • 13 min read

Front end interview (2 Part Series)

1 **All front end Interview questio…**

2 Resources for interview prepar…

Abhijeet Yadav

A software enthusiast with a passion for learning and exploring new things.

Follow

WORK
Software Engineer III at Walmart Global Tech India

LOCATION
Mumbai, india

JOINED
May 10, 2020

More from Abhijeet Yadav

Resources for interview preparations (Front End).

#javascript #html #css #react

Binding translations to views

#react #i18n #localisation #translation

Interview questions for front end

This readme is a compilation of all the question asked during my recent COVID-19 job hunt. I've also attached a list of resources that I'd referred for the preparations.

The questions are divided into following sections.

- JS
- Coding
- Assignments
- Miscellaneous

> The solution are not production ready code and just represents a rough idea of my approach. Try

Exposing API using React provider

#i18n #react #internationalization #localisation

> implementing your own approach.
>
> Edits made on 20/08/2020. Click to view changes

JS

1) Given a multidimensional array with depth of n, flatten it. Once flattened make it available as a method on `array` instance

Solution

```
/**
 * [1,2,[3,4]] -> [1,2,3,4]
 */

let arr = [1,2,[3,4, [5,6, [7, [

function flatten(arr) {
  return arr.reduce(function(acc
    let isArray =  Array.isArray
    return acc.concat(isArray ?
  }, [])
}

if (!Array.prototype.flatten) {
  Array.prototype.flatten = func
    return flatten(this)
  }
}
console.log(arr.flatten());
```

---

2) Create a promise from scratch

Solution

```
class CustomPromise {
  state = "PENDING"
  value = undefined
  thenCallbacks = []
  errorCallbacks = []

  constructor(action) {
    action(this.resolver.bind(th
  }

  resolver(value) {
    this.state = "RESOLVED"
    this.value = value
    this.thenCallbacks.forEach((
      callback(this.value)
    })
  }

  reject(value) {
    this.state = "REJECTED"
    this.value = value
    this.errorCallbacks.forEach(
      callback(this.value)
    })
  }

  then(callback) {
    this.thenCallbacks.push(call
    return this
  }

  catch (callback) {
    this.errorCallbacks.push(cal
    return this
  }
}

let promise = new CustomPromise(
  setTimeout(() => {
    const rand = Math.ceil(Math.
    if (rand > 2) {
      resolver("Success")
```

```
    } else {
      reject("Error")
    }
  }, 1000)
})

promise
  .then(function(response){
    console.log(response)
  })
  .catch(function(error){
    console.log(error)
  })
```

---

3) Filter movie list by average rating, name. Sort filtered list by any field inside movie object

Solution

```
// O(M)
function getMovies() {
  return []; // [{id, name, year
}

// O(R)
function getRatings() {
  return []; // [{id, movie_id,
}

/**
 * minAvgRating ->
 *    avgRating >= minAvgRating
 *
 * sort ->
 *    name -> ascending order mo
 *   -name -> descending
 *
 *    avgRating
```

```
 *
 *
 * search ->
 *   'ave' -> 'Avengers'
 *   'avengers' -> 'Avengers'
 *   'AvengersInfinitywar' -> 'A
 */
const toLower = str => str.toLoc

const getAvrgRating = (movie, mo
  let count = 0;
  return movingWithRatings.reduc
    const movieMatch = movie.id
    if (movieMatch) {
      acc+=value.rating
      count++
    }
    if (index === movingWithRati
      acc = acc/count
    }
    return acc
  }, 0)
}

const isSubString = (str1, str2)
  str1 = toLower(str1.split(" ")
  str2 = toLower(str2.split(" ")
  if (str1.length > str2.length)
    return str1.startWith(str2)
  } else {
    return str2.startWith(str1)
  }
}

const moviesList = getMovies()
const movingWithRatings = getRat
function queryMovies({ search, s
  let filteredMovies = movingWit
  filteredMovies = filteredMovie
  filteredMovies = filteredMovie
  filteredMovies = filteredMovie
    const isDescending = sort[0]
    let sortCopy = isDescending
```

```
    const value1 = a[sortCopy]
    const value2 = b[sortCopy]
    if (isDescending) {
      return value1 > value2 ? -
    }else {
      return value1 < value2 ? -
    }
  })
  filteredMovies = filteredMovie
    ...movie,
    avgRating: movingWithRatings
  }))
  return filteredMovies
}
```

---

4) Given an end point URL to fetch all the `posts` and `comments`. Do the following.

- Map all the comments to the posts it belongs to. The resultant data after mapping should be of below structure.

Solution

```
//service.js
const POSTS_URL = `https://jsonp
const COMMENTS_URL = `https://js

export const fetchAllPosts = ()
  return fetch(POSTS_URL).then(r
};

export const fetchAllComments =
  return fetch(COMMENTS_URL).the
};
```

```
import { fetchAllPosts, fetchAll



const fetchData = async () => {
  const [posts, comments] = awai
    fetchAllPosts(),
    fetchAllComments()
  ]);

  const grabAllCommentsForPost =
    comments.filter(comment => c

  const mappedPostWithComment =
    const allComments = grabAllC
    acc[post.id] = allComments;
    return acc;
  }, {});

  console.log("mappedPostWithCom
};

fetchData();
```

---

5) Implement a method `getHashCode` on string instance. The method should be available on all strings.

Solution

```
let s1 = "sample"

if (!String.prototype.getHashCod
  String.prototype.getHashCode =
    console.log('String instance
    return this
  }
}
```

6) What does the below expressions evaluate to

```
1+true
true+true
'1'+true
'2' > '3'
'two'>'three'
```

Solution

```
2
2
1true
false
true
```

7) Implement `bind` and `reduce`.

Solution

```
//bind
if (!Function.prototype.bind) {
  Function.prototype.bind = func
    const func = this
    const context = arg[0]
    const params = arg.slice(1)
    return function(...innerPara
      func.apply(context, [...pa
    }
  }
}
```

```
//reduce
Array.prototype.reduce = functio
  const arr = this
  const callback = func
  let init = initState

  arr.forEach(function(value, in
      init=callback(init, value)
  })
  return init
}
```

---

8) Implement debounce function

Solution

```
const debounce = function(func,
  let timerId;
  return function(e){
    clearTimeout(timerId)
    timerId = setTimeout(functio
      func.apply()
    }, interval)
  }
}
debounce(apiCall, 3000)
```

---

9) Implement throttling function

Solution

```
const throttle = (callback, inte
  let timerId;
  let allowEvents = true;

  return function() {
    let context = this;
```

```
      let args = arguments;

      if (allowEvents) {
        callback.apply(context, ar
        allowEvents = false;
        timerId = setTimeOut(funct
          allowEvents = true
        }, interval)
      }
    }
  }
```

---

10) Design API polling mechanism. The API is called after a fixed interval. The API is a stock API that fetches the latest price of stock. Upon fetching the results, render the UI.

The question demands the design aspect of the solution and not the code. It was open ended question.

Solution

```
//With setInterval, throttling a
setInterval=>Endpoint=>Render

//with the inversion of control
//Endpoint=>Render=>setTimeout=>
```

---

11) Convert class based inheritance code given below to ES5 code.

```
class Parent(name){
  constructor(name) {
```

```
    this.name=name
  }

  getName(){return this.name}
}

class Children extends Parent {
  constructor(props){
    super(props)
  }
}
```

Solution

```
function Parent(name) {
  this.name = name
}

Parent.prototype.getName = funct
  return this.name
}

function Children(name){
  Parent.call(this, name)
}

Children.prototype = new Parent(
```

---

12) What does following code evaluates to?

```
//Q.1
var x = 1;
var y = x;

x = 0;
console.log(x, y);
```

```
//Q.2
var x = [1];
var y = x;

x = [];
console.log(x,y);

//Q.3
function Abc() { console.log(thi
Abc()
new Abc();

//Q.4
var x = 1;
var obj = {
  x: 2,
  getX: function () {
    return console.log(this.x);
  }
};

obj.getX()
let a = obj.getX
console.log(a)

//Q.5
//How to get the a to log 2 in t

//Q.6
console.log("A");
setTimeout(() => console.log("B"
setTimeout(() => console.log("C"
console.log("D");

//Q.7
setTimeout(function() {
  console.log("A");
}, 0);
Promise.resolve().then(function(
  console.log("B");
}).then(function() {
  console.log("C");
});
```

```
console.log("D");

//Q.8
let obj1 = {
  a:1,
  b:2
}

function mutate(obj) {
  obj = {a:4, c:6}
}

console.log(obj1)
mutate(obj1)
console.log(obj1)
```

Solution

```
//A.1
0 1

//A.2
[] [1]

//A.3
window object is logged

//A.4
logs 2 and 1

//A.5
a.call(obj);

//A.6
A, D, B , C

//A.7
D, B, C, A

//A.8
```

```
{ a: 1, b: 2 }
{ a: 1, b: 2 }
```

---

13) Given an array of numbers implement the following

```
const list = [1,2,3,4,5,6,7,8]
const filteredArray = list.filte
```

Solution

```
function between(start, end) {
  return function (value,index)
    return value>start && value<
  }
}
```

Algorithms

1) Consider the following series:

```
A := 1
B := A*2 + 2
C := B*2 + 3 and so on...
```

Write a program that:

outputs the number corresponding to a given letter

given a string of letters like 'GREP', computes the sum of the numbers corresponding to all the letters in the

string (i.e., G + R + E + P), as given by the above series and

given a large number (that would fit into a standard 32-bit integer), finds the shortest string of letters corresponding to it.

You may use a greedy approach for the last part. Compute the values of the numbers corresponding to letters as and when required and DO NOT pre-compute beforehand and store them in a data structure.

Solution

```
//A = 1
//B = A*2 +2
//C = B*2+ 3
//D = C*2+ 3

var genCharArray = function(char
    var a = [], i = charA.charCo
    for (; i <= j; ++i) {
        a.push(String.fromCharCo
    }
    return a;
}

var charMap = {};
var charArray = genCharArray('a'

charArray.forEach(function(char,
    charMap[char] = Number(index
});


var charSequence = function(char
    if(typeof char==="string"){
```

```
            char = charMap[char];
        }
        if(char==1){
            return 1;
        }else{
            return char + 2 * charSe
        }
}

var input = process.argv[2];

if(input.length===1){
    console.log(charSequence(cha
}else if(input.length>1){
    var charTotalSequence = inpu
        return acc + charSequenc
    },0);
    console.log(charTotalSequenc
}
```

---

2) Given an array find a pair such that it sums to a given number

Solution

```
let nums = [2, 7, 10, 1, 11, 15,
let target = 11
let numsMap = new Map()
let pairs = nums.reduce((acc, nu
  let numToFind = target - num
  if (numsMap.get(numToFind)) {
    return [...acc, [num, numToF
  } else {
    numsMap.set(num, true)
    return [...acc]
  }
}, [])

console.log("Pairs ", pairs)
```

3) Find the local maxima in a given array. A local maxima is a element that is greater than it's left and right neighbours. I provided a O(n) solution which was quite straight forward before going for optimisation.

Solution

```
let x = [1, 2, 3, 5, 4] //Output
if x.length == 1 return x[0]
else
 let i = 1
 for(;i<x.length-1;i++){
  if x[i-1]<x[i] and x[i] > x[i+
 }
 if x.length - 1 == i return x[i
```

4) Rotate a matrix clockwise by 90 degree. The solution should be in place.

leetcode

Solution

```
[
 [1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]
]
//The solution is to first take
//After taking the transpose the
[
 [1, 4, 7],
```

```
 [2, 5, 8],
 [3, 6, 9]
]
//After the transpose step, All
//The resulting matrix after aft
[
 [7, 4, 1],
 [8, 5, 2],
 [9, 6, 3]
]
//The above matrix is rotated 90
```

---

5) Maximum subarray sum modulo m

Solution

---

6) Given an array find three element in array that sum to a given target

Solution

```
let x = [1, 2, 3, 4, 5]
let target = 7
let found = []

const twoPointer = (l ,r, curren
  while(l<r){
    const totalSum = current + x
    if (totalSum === target) {
      found.push([current, x[l],
      return
    } else if (totalSum > target
      r--
    } else {
      l++
    }
  }
}
```

```
const threeSum = (x, target) =>
    for (let i=0;i<x.length;i++)
      const current = x[i];
      let leftPointer = i+1
      let rightPointer = x.lengt

      if (current+x[leftPointer]
        found.push([current, x[1
      } else {
        twoPointer(leftPointer,
      }
  }
  return found
}
```

---

7) Given a string and an integer k, find number of substrings in which all the different characters occurs exactly k times.

link

Solution

```
const subStrHasSameCharCount = (
  let charMap = {}
  for (let k=startIndex;k<endInd
    let currentChar = str[k]
    if (charMap[currentChar]) {
      charMap[currentChar]++
    } else {
      charMap[currentChar] = 1
    }
  }
  let totalCount = Object.values
  return totalCount ? Object.val
}
```

```
const characterWithCountK = (str
  if (k == 0) return ''
  let count = 0
  let initialHop = k
  while (initialHop < str.length
    for (let j=0;j<str.length;j+
      let startIndex = j
      let endIndex = j + initial
      if(endIndex > str.length)
      count = subStrHasSameCharC
        ? count + 1: count
    }
    initialHop+=k
  }
  count = subStrHasSameCharCount
        ? count + 1: count
  return count
}


let str = 'aabbcc'
let k = 2
console.log(characterWithCountK(
```

8) Given two input strings, s1 and s2, containing characters from a-z in different orders, find if rearranging string in s1 results in a string that is equal to s2.

Solution

```
let s1 = 'dadbcbc'
let s2 = 'ccbbdad'
let charMap = {}

const canBeRearranged = (s1, s2)
  if(s1.length!==s2.length){
```

```
    return false
  }
  for(let i=0;i<s1.length;i++){
    const charFromString1 = s1[i
    const charFromString2 = s2[i
    if(charFromString1 in charMa
      charMap[charFromString1]++
    } else {
      charMap[charFromString1] =
    }
    if(charFromString2 in charMa
      charMap[charFromString2]--
    } else {
      charMap[charFromString2] =
    }
  }
  for(let x in charMap){
    if (charMap[x]!==0){
      return false
    }
  }
  return true
}

canBeRearranged(s1, s2)
```

---

9) Given an array or variable input size, write a function to shuffle the array.

Solution

```
const swap = (index1, index2, ar
  let temp = arr[index1]
  arr[index1] = arr[index2]
  arr[index2] = temp
}

const shuffle = (arr) => {
```

```
  let totalLength = arr.length
  while(totalLength > 0) {
    let random = Math.floor(Math
    totalLength--
    swap(totalLength, random, ar
  }
  return arr
}


let arr = [1, 2, 3, 4, 5, 6, 7,
arr = shuffle(arr)
```

---

10) Calculate the sum of all elements in a multidimensional array of infinite depth.

Solution

```
let arr = [4, 5, 7, 8, [5, 7, 9,
let sum = 0

const calculateSum = (arr) => {
  arr.reduce(function(acc, curre
    const isEntryArray = Array.i
    if (isEntryArray) {
      return acc.concat(calculat
    } else {
      sum+=currentVal
      return acc.concat(currentV
    }
  }, [])
}
calculateSum(arr)
console.log(sum)
```

---

11) Flatten a nested object of varying debt.

Solution

```
const obj = {
  level1: {
    level2: {
      level3: {
        more: 'stuff',
        other: 'otherz',
        level4: {
          the: 'end',
        },
      },
    },
    level2still: {
      last: 'one',
    },
    am: 'bored',
  },
  more: 'what',
  ipsum: {
    lorem: 'latin',
  },
};

var removeNesting = function(obj
  for (let key in obj){
    if (typeof obj[key] === "obj
      removeNesting(obj[key], pa
    } else {
      flattenedObj[parent+'.'+ke
    }
  }
}

let flattenedObj = {}
const sample = removeNesting(obj
console.log(flattenedObj);
```

12) Given a json input, where each entry represents a directory such that each directory in turn can have a nested entry of it's own. Create the resulting directory structure.

Solution

---

13) Given an array of object containing list of employee data such that each employee has list of reportee. Use this information to construct a hierarachy of employees.

Solution

```
const employeesData = [{
  id: 2,
  name: 'Abhishek (CTO)',
  reportees: [6]
}, {
  id: 3,
  name: 'Abhiram (COO)',
  reportees: []
}, {
  id: 6,
  name: 'Abhimanyu (Engineering
  reportees: [9]
}, {
  id: 9,
  name: 'Abhinav (Senior Enginee
  reportees: []
}, {
  id: 10,
  name: 'Abhijeet (CEO)',
  reportees: [2, 3],
}];

/*
```

```
A (CEO)
----B (CTO)
--------D (Engineering Manager)
------------E (Senior Software E
----C (COO)
*/

const findCeo = (currentEmp) =>
  let parentEmployee = employees
  if (parentEmployee && parentEm
    return findCeo(parentEmploye
  } else {
    return currentEmp
  }
}

const logHierarchy = (currentEmp
  console.log("-".repeat(indent)
  indent+=4;
  for(let i=0;i <currentEmp.repo
    let employee = employeesData
    logHierarchy(employee[0], in
  }
}

const traverse = (employee) => {
  let ceo = findCeo(employee)
  logHierarchy(ceo, 0)
}

traverse(employeesData[0])
```

---

14) Print a given matrix in spiral form

```
const inputMatrix = [
  [1, 2, 3, 4,  5],
  [6, 7, 8, 9, 10],
  [11,12,13,14,15],
  [16,17,18,19,20],
]
```

```
const exprectOutput = [1,2,3,4,5
```

Solution

```
function spiralParser(inputMatri
  const output = [];
  let rows = inputMatrix.length;
  let cols = rows > 0 ? inputMat

  //singleEmptyRow => Edge case
  if (rows === 0) {
    return []
  }

  if (rows === 1) {
    //singleElementRowNoCol => E
    if (cols === 0) {
      return []
    } else if (cols === 1){
      //singleElementRow => Edge
      output.push(inputMatrix[0]
      return output
    }
  }

  let top = 0;
  let bottom = rows - 1;
  let left = 0;
  let right = cols - 1;
  let direction = 0;
  //0 => left->right
  //1 => top->bottom
  //2 => right->left
  //3 => bottom->top

  while(left <= right && top <=
    if(direction === 0) {
      //left->right
      for (let i=left; i<=right;
        output.push(inputMatrix[
```

```
      }
      top++;
    } else if (direction === 1)
      //top->bottom
      for (let i=top; i<=bottom;
        output.push(inputMatrix[
      }
      right--
    } else if (direction === 2)
      //right->left
      for (let i=right; i>=left;
        output.push(inputMatrix[
      }
      bottom--
    } else if (direction === 3)
      //bottom->top
      for (let i=bottom; i>=top;
        output.push(inputMatrix[
      }
      left++
    }
    direction = (direction + 1)
  }
  return output;
}

console.log(spiralParser(inputMa
```

---

15) Find maximum consecutive repeating char in a give string.

```
let str = 'bbbaaaaccadd'; //max
```

Solution

```
//sudo code
maxNow = if input string length
maxOverall = if input string len
```

```
for char in inputString starting
  if char equals prevChar
    maxNow++
    maxOverall = max(maxOverall,
  else if char not equals prevCh
    maxNow = 1
```

---

16) Given a input array of varying length, segregate all the 2's at the end of the array.

```
let inputArr = [2,9,1,5,2,3,1,2,
//ouput => [9,1,5,3,1,7,4,3,8,29
```

Solution

```
let slowRunner = 0

for (let fastRunner=0;fastRunner
  if (arr[fastRunner]!==2 && arr
    [arr[fastRunner], arr[slow]]
    slowRunner++
  }
}
```

---

17) Reverse a linked list

```
//Input = 1 -> 2 -> 3 -> 4 -> 5
//Output = 1 <- 2 <- 3 <- 4 <- 5
```

Solution

```
//sudo code
let current = head
let prev = null
let next = null

while(current) {
  next = current.next
  current.next = prev
  prev = current
  current = next
}
```

---

18) preorder tree traversal using iteration (No recurssion)

Solution

```
//sudo code
const preorder = (root) => {
  let stack = []
  stack.push(root)

  while(there is element in stac
    let current = stack.pop()
    console.log(current.value)
    if (current.right) {
      stack.push(current.right)
    }
    if (current.left) {
      stack.push(current.left)
    }
  }
}
```

Assignments

1) Design a parking lot system with following requirements:

- It can hold up to N vehicles. Handle availability of parking lot.
- Entry and exit log of vehicles.
- Automated ticketing system for every vehicle entering/exiting the parking lot will have vehicle registration with vehicle details like: Registration No., Color, allocated parking slot.

I should be able to query:

- Registration No. of all vehicles of a particular color.
- Parking slot of a vehicle given registration no.
- Parking slots for vehicles given a color.
- List of available slots in the parking lot.

Requirements:

- Can use anything to structure the code: Classes/Structs.
- Your solution should be extendable for future use cases.

Few code design principles:

- Modularity of code.
- Naming conventions.

- SOLID principles.

Solution

---

2) Create a react component `Ping` that makes an API call to a given URL. If the API calls returns status code 200 that means the user is online. However, if the API call receives status code other than 200 it means, the user is offline.

> Try changing `status` form dev tools network panel

Solution

---

3) Create a dynamic form builder from a `json` input. The form can be grouped based on `id`. Each group can have a nested group of it's own.

Solution

---

4) Create a minimal excel sheet in pure javascript that supports `adding` and `removing` rows, columns. There was time limit on this question of 40 minutes.

Solution

---

5) You have to make a search input box which will search over a list of users.

The user object has the following fields

```
id: a unique id
name: user's name
items: list of items ordered
address: address of the user
pincode: user address pin cod
```

You have to implement search on all of the fields.

The search results will show up as a list of User Cards.

To Summarize
On typing in the search input box, the search results list opens up. The search could be just a string matching search.

The list of cards can be navigated through keyboard or mouse
only one card should highlight at a time if both mouse and keyboard are used for navigation

(keyboard will take preference if mouse is kept hovered on the list, similarly mouse will take preference if keyboard navigation is not used).

This behaviour is similar to how youtube search works

When no search results are found, an empty card is displayed
The card list would be scrollable.

The highlighted card (via keyboard/mouse) will scroll into view

Solution

Miscellaneous

1) How would you architect a front end application click
1) Implement Lazy loading click
2) What is Server Side Rendering.
3) How to deploy a react app to production.
4) What are service worker/web worker.
5) How to optimise a web app and make it more performant.
6) Explain different type of client side cache strategies.
7) What is CORS.
8) What are higher order component in react.
9) How does connect function work in redux.
10) What are pure components in React.

Resources

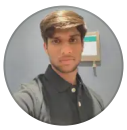
link

Front end interview (2 Part Series)

1 **All front end Interview questio…**

2 Resources for interview prepar…

Posted on Aug 4 by:



Abhijeet Yadav

@devabhijeet

A software enthusiast with a passion for learning and exploring new things.

Follow

Discussion

Subscribe

Add to the discussion

PREVIEW SUBMIT

ogrotten Aug 5

This list of problems to solve is ***extremely*** discouraging. I've done Hackerrank/leetcode/codesignal challenges, and most of these seem like

things that would take me *at least* a couple hours. Mainly, *I can't even see* a 'first step' to a solution after reading the problem. But if you have all these exercises, it means you had all these interviews without being hired... and if they didn't hire you, then what gnats chance in hell do i have?

Congratulations, truly, for getting the interviews, and I thank you kindly for posting these in an attempt to help others learn from your experience.

3 REPLY

Abhijeet Yadav Aug 5

Yeah, I know right. It's not like I had it all figured out. The solutions I have posted here are the final ones that I had reached after making countless mistake and a ton of console logs. The toughest amongst them were 1 and 7. On any other day I wouldn't be struggling that much but presenting solution during interview is a whole different ball game.

Also I had cleared a lot of interviews but most of it didn't materialize. Some of the interviews I gave were only to gain experience, even when I was not interested in the company I was getting interviewed at.

1 REPLY

John Mendez Sep 17

These interview questions aren't necessarily meant to be answered on the spot. They're purposely complex so that the interviewer can see your thought and team process.

I was dumbfounded by many interview questions and I'm currently in a few final rounds. One of them is a fortune 500 company, and that's the one I most

shaky in and never wrote a full implementation. But by talking my way through it aloud, the interviewer realized that I knew what I was doing and needed more time.

Just make sure to collaborate with the interviewer as if you already work there. Ask questions and provide input as you search for the solution. Understand why you're making certain choices, and the tradeoffs of them. They want to know how you'll be on the job after all.

Any honest developer would agree that if you haven't seen these problems before, it would take more than a couple of hours.

Like, who seriously believes promises were implemented in a day, let alone a couple of hours? And for those who do think that, do you even want to work with that toxic mentality?

The path to the solution is more important than the solution. So focus less on interviewing, and more on acting the same as you would on the job.

1 REPLY

ogrotten Sep 17

Solid take and advice. Thanks.

2 REPLY

asiraky Aug 7

I disagree. These questions are not that hard for someone calling themself a software engineer.

1 REPLY

ogrotten Aug 7

Very helpful. Congratulations on your new title.

6 THREAD

asiraky Aug 7

A tad unnecessary. Just stating my opinion.

1 REPLY

Alejandro esquivel Aug 10

I disagree with your disagreement, this post has the tag react, which is a front-end web dev library, not a software engineering thing.

1 THREAD

asiraky Aug 10

Never mind how little sense that makes, the author actually stated he was being interviewed for a Software Engineer position. How is the tag of the post relevant, given that fact.

1 REPLY

रोहन मल्होत्रा Aug 10

Your promise implementation is incorrect.
This is how the promise works:
If the handler function:

- returns a value, the promise returned by then gets resolved with the returned value as its value.
- doesn't return anything, the promise returned by then gets resolved with an undefined value.

- throws an error, the promise returned by then gets rejected with the thrown error as its value.
- returns an already fulfilled promise, the promise returned by then gets fulfilled with that promise's value as its value.
- returns an already rejected promise, the promise returned by then gets rejected with that promise's value as its value.
- returns another pending promise object, the resolution/rejection of the promise returned by then will be subsequent to the resolution/rejection of the promise returned by the handler. Also, the resolved value of the promise returned by then will be the same as the resolved value of the promise returned by the handler.

4 REPLY

Abhijeet Yadav Aug 10

The implementation is a rough idea.

1 REPLY

vibhanshu pandey Aug 14

I believe it's better to provide a correct solution or no solution at all, otherwise, it may mislead our fellow developers into believing it to be the correct solution. You may check out this link for a detailed implementation and explanation of promises. By the way great post. :)

1 THREAD

Abhijeet Yadav Aug 14

Please read the note before the answer started.

vibhanshu pandey Aug 15 THREAD

Yes, I've read that, it says **"not production ready"**, not that the provided solutions are psudo-code. Anyway great stuff 👍.

1 REPLY

Kevin ALBRECHT Aug 7

Hi, thank you for sharing this but can you explain your getHashCode implementation ?
I though you would compute some values, based on charCodeAt() and maybe bitwise operation or anything, to make it unique, but you simply returned the string instance.

1 REPLY

Abhijeet Yadav Aug 7

I expected the same when the interviewer asked the question. However, on further follow up the interviewer asked me to only make a method by the same name available on the instance. Strange 😕

1 REPLY

Kevin ALBRECHT Aug 7

Oh ok weird, thank you.
great blogpost anyway :)

1 REPLY

Konstantin Kovalev Aug 21

2) Create a promise from scratch

Sorry, but your promise implementation is wrong. `then` and `catch` must return new `Promise` instance but not `this`. That's why your sample doesn't work correct:

```
promise
  .then(function(response){
    console.log(response);
  return [...response];
  }).
then(response => {
  console.log(response);
})
  .catch(function(error){
    console.log(error)
  });
```

Compare results using original `Promise`.

1 REPLY

bobgravity1 Aug 7

lol. here i am a year into learning JS about an hour a day just mainly for fun. got more serious recently about trying for an entry level front end job and have been applying about 3 hours a day. but after seeing this im crushed... yay! thanks for sharing lol

1 REPLY

kurtyazdizadeh Aug 5

What types of positions did you apply for? Entry-level? Junior? Mid? Senior level? Just curious!

1 REPLY

Abhijeet Yadav Aug 6

It was Software Engineer III position.

2 REPLY

 Joshua Sep 16

That makes sense

1 REPLY

 Ezell Frazier Aug 5

Absolute insanity.

2 REPLY

 Pankaj Jaiswal Aug 10

All this stuff for Frontend post? I am a fresher starting career in Frontend Development. But I hate DSA from my bottom of the heart. What should I do? These types of questions will freak me out.

1 REPLY

 Abhijeet Yadav Aug 21

I am not sure if a fresher will be subjected to same questions. However, learning DSA will help you out in long term if you are considering Software development for profession. You can check my `Resources for preparation` blog. It has links to some youtube channel to get started with DSA.

1 REPLY

 bobgravity1 Aug 7

is this type of stuff way beyond a junior front end web dev interview? i f****in hope

so lol... i would NEVER be able to answer this

1 REPLY

 Michael Lee Aug 7

Why does 'two' > 'three' evaluate to true?

1 REPLY

 Abhijeet Yadav Aug 7

For this case, ASCII value of first character from both string are compared. Since the first char is 't' from both string, the resultant ASCII value will also be same. The evaluation then moves to the second character of both string.

Character 'w' from string one has ASCII value greater than ASCII value of character 'h' from string two, hence evaluation will now be of below state...

```
ASCII(w) > ASCII(h) => true
```

Also the below evaluates to false

```
'tWo' > 'three' //false
```

1 REPLY

Comment marked as low quality/non-constructive by the community **View code of conduct**

 Sriram Aug 4

Great work I'm sriram fullstack javascript developer I have a whatsapp group dedicated to coders so that you can chat and collaborate on fun hobby projects with

real people if you are interested please ping me at +918970787208

1 REPLY

Code of Conduct • Report abuse

Read next

Test Driven Development using Cypress

Amit Kumar Das - Aug 21

Async/await can still surprise you... A LOT!

Charles Assunção - Aug 19

What is Closure in JavaScript?

Brett Fisher - Aug 20

Building git part-1

Rafi - Aug 20

Built on Forem — the open source software that powers DEV and other inclusive communities.

Made with love and Ruby on Rails.