TestDome Js interview questions:

**JavaScript Interview Questions & Answers**

**Question 1: Ensure**

Implement the ensure function so that it throws an error if called without arguments or the argument is undefined. Otherwise it should return the given value.

Starting code:

function ensure(value) {

}

Solution:

function ensure(value) {

if (value !== undefined) return value;

throw 'Error';

}

**Question 2: Remove Property**

Implement the removeProperty function which takes an object and property name, and does the following:

If the object obj has a property prop, the function removes the property from the object and returns true; in all other cases it returns false.

Starting code:

function removeProperty(obj, prop) {

return null;

}

Solution:

function removeProperty(obj, prop) {

if(Object.keys(obj).indexOf(prop) >-1){

delete obj[prop];

return true;

}

else{

return false;

}

}

**Question 3: Check Digit**

Your company assigns each customer a membership ID, and you are implementing a check digit for those IDs.

The check digit should be calculated by adding up all digits in each membership ID. If the result of the sum is a number with more than a single digit, another iteration is required, and the digits of the result also should be added together. This process should repeat until a single-digit number is calculated.

For example, for the membership ID "55555" the sum of all digits is 25. Because this is not a single-digit number, 2 and 5 would be added, and the result, 7, would be the check digit.

Starting code:

/\*\*

\* @prop {string} membershipId: The customer's membership ID.

\* @return {number} The check digit.

\*/

function createCheckDigit(membershipId) {

// Write the code that goes here.

return 0;

}

console.log(createCheckDigit("55555"));

Solution:

function createCheckDigit(membershipId) {

// Write the code that goes here.

if(membershipId.length>1){

var final =adder(membershipId);

return final;

}

else return membershipId;

};

function adder(membershipId){

var arr=membershipId.split("");

var sum;

if( arr.length>1){

sum= arr.reduce(function myfunction(total,currentvalue){

return parseInt(total)+ parseInt(currentvalue);

},0);

}

if(sum>9){

return adder(sum.toString());

}

else return sum;

}

console.log(createCheckDigit("55555"));

/\*\*

\* @prop {string} membershipId: The customer's membership ID.

\* @return {number} The check digit.

\*/

function createCheckDigit(membershipId) {

let sumMembershipId = aggregator(membershipId);

while (parseInt(sumMembershipId) > 9) sumMembershipId = aggregator(sumMembershipId);

return sumMembershipId;

}

function aggregator(strMembershipId) {

return strMembershipId.toString().split('').reduce((a, b) => parseInt(a) + parseInt(b), 0);

}

console.log(createCheckDigit("55555"));

**Question 4: Date**

Write a function that converts user entered date formatted as M/D/YYYY to a format required by an API (YYYYMMDD). The parameter "userDate" and the return value are strings.

For example, it should convert user entered date "12/31/2014" to "20141231" suitable for the API.

Starting code:

function formatDate(userDate) {

// format from M/D/YYYY to YYYYMMDD

}

console.log(formatDate("12/31/2014"));

Solution:

function formatDate(userDate) {

// format from M/D/YYYY to YYYYMMDD

var x=userDate.split('/',3);

var x1=x[2];

var x2= ((x[0].length ==2)? x1.concat("",x[0]):x1.concat("",0,x[0]));

var res= ((x[1].length ==2)? x2.concat("",x[1]):x2.concat("",0,x[1])) ;

return res;

}

console.log(formatDate("12/31/2014"));

**Question 5: Image Gallery**

An image gallery is a set of images with corresponding remove buttons. This is the HTML code for a gallery with two images:

<div class="image">

<img src="https://goo.gl/kjzfbE" alt="First">

<button class="remove">X</button>

</div>

<div class="image">

<img src="https://goo.gl/d2JncW" alt="Second">

<button class="remove">X</button>

</div>

Implement the setup function that registers a click event handler and implements the following logic: When the button of class remove is clicked, its parent

element should be removed from the gallery.

For example, after the first image has been removed from the gallery above, it's HTML code should look like this:

<div class="image">

<img src="https://goo.gl/d2JncW" alt="Second">

<button class="remove">X</button>

</div>

Starting code:

function setup() {

// Write your code here.

}

// Example case.

document.body.innerHTML = `

<div class="image">

<img src="https://goo.gl/kjzfbE" alt="First">

<button class="remove">X</button>

</div>

<div class="image">

<img src="https://goo.gl/d2JncW" alt="Second">

<button class="remove">X</button>

</div>`;

setup();

document.getElementsByClassName("remove")[0].click();

console.log(document.body.innerHTML);

Solution:

function setup() {

var items = document.getElementsByClassName("remove");

for (let i = 0; i < items.length; i++) {

items[i].addEventListener('click', function removeParentDiv() {

this.parentNode.remove();

});

}

}

// Example case.

document.body.innerHTML = `

<div class="image">

<img src="https://goo.gl/kjzfbE" alt="First">

<button class="remove">X</button>

</div>

<div class="image">

<img src="https://goo.gl/d2JncW" alt="Second">

<button class="remove">X</button>

</div>`;

setup();

document.getElementsByClassName("remove")[0].click();

console.log(document.body.innerHTML);

**Question 6: Closures**

Fix the bugs in the registerHandlers function. An alert should display anchor's zero-based index within a document instead of following the link.

For example, in the document below, the alert should display "2" when Google anchor is clicked since it is the third anchor element in the document and its zero-based index is 2.

<body>

In my life, I used the following web search engines:<br/>

<a href="//www.yahoo.com">Yahoo!</a><br/>

<a href="//www.altavista.com">AltaVista</a><br/>

<a href="//www.google.com">Google</a><br/>

</body>

Starting code:

function registerHandlers() {

var as = document.getElementsByTagName('a');

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

as[i].onclick = function() {

alert(i);

return false;

}

}

}

/\* HTML code for testing purposes (do not submit uncommented):

<body>

In my life, I used the following web search engines:<br/>

<a href="//www.yahoo.com">Yahoo!</a><br/>

<a href="//www.altavista.com">AltaVista</a><br/>

<a href="//www.google.com">Google</a><br/>

</body>

\*/

Solution:

function registerHandlers() {

var as = document.getElementsByTagName('a');

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

as[i].onclick = function(i) {

return function() {

alert(i);

return false;

}

}(i);

}

}

/\* HTML code for testing purposes (do not submit uncommented):

<body>

In my life, I used the following web search engines:<br/>

<a href="//www.yahoo.com">Yahoo!</a><br/>

<a href="//www.altavista.com">AltaVista</a><br/>

<a href="//www.google.com">Google</a><br/>

</body>

\*/

**Question 7: Loop**

Function appendChildren should add a new child div to each existing div. New divs should be decorated by calling decorateDiv.

For example, after appendChildren is executed, the following divs:

<div id="a">

<div id="b">

</div>

</div>

should take the following form (assuming decorateDiv does nothing):

<div id="a">

<div id="b">

<div></div>

</div>

<div></div>

</div>

Starting code:

function appendChildren(decorateDivFunction) {

var allDivs = document.getElementsByTagName("div");

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

var newDiv = document.createElement("div");

decorateDivFunction(newDiv);

allDivs[i].appendChild(newDiv);

}

}

// Example case.

document.body.innerHTML = `

<div id="a">

<div id="b">

</div>

</div>`;

//appendChildren(function(div) {});

console.log(document.body.innerHTML);

Solution:

function appendChildren(decorateDivFunction) {

var allDivs = [...document.getElementsByTagName("div")];

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

var newDiv = document.createElement("div");

decorateDivFunction(newDiv);

allDivs[i].appendChild(newDiv);

}

}

// Example case.

document.body.innerHTML = `

<div id="a">

<div id="b">

</div>

</div>`;

// appendChildren(function(div) {});

console.log(document.body.innerHTML);

**Write a function that can determine whether a string is a palindrome in under 100 characters.**

**A**

A palindrome is a word, phrase, or sequence of letters that reads the same backwards or forwards. It also makes a great test for checking their ability to handle strings.

function isPalindrome(str) {

str = str.replace(/s/g, '').toLowerCase();

return (str == str.split('').reverse().join(''));

}

**Determine the output of the code below. Explain your answer.**

**A**

var myObject = {

egg: “plant”,

func: function() {

var self = this;

console.log(“outer func: this.egg = “ + this.egg);

console.log(“outer func: self.egg = “ + self.egg);

(function() {

console.log(“inner func: this.egg = “ + this.egg);

console.log(“inner func: self.egg = “ + self.egg);

}());

}

};

myObject.func();

This question is designed to test the interviewee’s understanding of scope and the “this” keyword. In the outer function, both “this” and “self” correctly refer to “myObject” and can subsequently access “egg.” In the inner function, “self” remains within scope while “this” can no longer refer to “myObject”—resulting in the output below:

outer func: this.egg = plant

outer func: self.egg = plant

inner func: this.egg = undefined

inner func: self.egg = plant

**Determine the output of the code below. Explain your answer.**

**A**

console.log(0.1 + 0.2);

console.log(0.4 + 0.1 == 0.5);

This is a trick question in that at first glance, you might expect the console to print out “0.3” and “true.” The correct answer is that you can’t know for sure, because of how JavaScript treats floating point values. In fact, in the above example, it will print out:

0.30000000000000004

false

**How would you empty the array below?**

**A**

Var emptyArray = [‘this’, ‘array’, ‘is’, ‘full’];

This deceptively simple question is designed to test your prospective coder’s awareness of mitigating potential bugs when solving problems. The easiest method would be to set “emptyArray” equal to “[ ]”—which creates a new empty array. However, if the array is referenced anywhere else, the original array will remain unchanged. A more robust method would be “emptyArray.length – 0;”—which not only clears the array but updates all reference variables that point to this original array. Some possible solutions are listed below:

emptyArray.length = 0;

emptyArray.splice(0, emptyArray.length);

while(emptyArray.length){

emptyArray.pop();

}

emptyArray = []

**Determine the output of the code below. Explain your answer.**

**A**

var lorem = { ipsum : 1};

var output = (function(){

delete lorem.ipsum;

return lorem.ipsum;

})();

console.log(output);

The output would be undefined, because the delete operator removed the property “ipsum” from the object “lorem” before the object was returned. When you reference a deleted property, the result is undefined.

Tuesdays afternoon – 38 regent st

519 church st – wednes afternoon

Church public school

<https://www.interviewcake.com/javascript-interview-questions>

<https://www.toptal.com/javascript/interview-questions>

**What will be the output of the code below?**

var Y = 1;

if (function F(){})

{

y += Typeof F;</span>

}

console.log(y);

|  |  |
| --- | --- |
|  |  |

The output would be 1undefined. The if condition statement evaluates using eval, so eval(function f(){}) returns function f(){} (which is true). Therefore, inside the if statement, executing typeof f returns undefined because the if statement code executes at run time, and the statement inside the if condition is evaluated during run time.

var arrayOfObjects = [

{

name: 'Doris',

num: 4

},

{

name: 'Beyonce',

num: 2

},

{

name: 'Albert',

num: 2

},

{

name: 'Diana',

num: 1,

}

];

function sortByMultipleKey(keys) {

return function(a, b) {

if (keys.length == 0) return 0; // force to equal if keys run out

key = keys[0]; // take out the first key

if (a[key] < b[key]) return -1; // will be 1 if DESC

else if (a[key] > b[key]) return 1; // will be -1 if DESC

else return sortByMultipleKey(keys.slice(1))(a, b);

}

}

arrayOfObjects.sort(sortByMultipleKey(['num', 'name']));

function multiply(a) {  
 return (b) => {  
 return (c) => {  
 return a \* b \* c  
 }  
 }  
}log(multiply(1)(2)(3)) // 6

Fizzbuzz problem

for(let i=0;i<100;)console.log((++i%3?'':'fizz')+(i%5?'':'buzz')||i)

<https://codeburst.io/javascript-breaking-down-the-shortest-possible-fizzbuzz-answer-94a0ad9d128a>