Multi Choice Type Question What is the output of the following JavaScript snippet: 123456789 let empids = "13218, 10535, 87456, 45156, 54544";let deptids = "745, 541, 621, 228, 967";let s1 = empids.charCodeAt(10);let s2 = deptids.charAt(5);let s3 = deptids.charCodeAt(65535);console.log(s1);console.log(s2);console.log(s3); Marks : 1 Negative Marks : 0 image Answer here 51 5 null 5 54544 NaN 51 5 NaN 5 54544 null

**ChatGPT said:**

Let's carefully evaluate the JavaScript snippet **step by step**:

let empids = "13218, 10535, 87456, 45156, 54544";

let deptids = "745, 541, 621, 228, 967";

let s1 = empids.charCodeAt(10);

let s2 = deptids.charAt(5);

let s3 = deptids.charCodeAt(65535);

console.log(s1);

console.log(s2);

console.log(s3);

**Step 1: s1 = empids.charCodeAt(10)**

* empids = "13218, 10535, 87456, 45156, 54544"
* The charCodeAt() method returns the Unicode of the character at the specified index.
* Index 10:

Index: 0 1 2 3 4 5 6 7 8 9 10

Chars: '1' 3 2 1 8 , 1 0 5 3 5

* Index 10 → character '5'
* Unicode (ASCII) of '5' = **53**

✅ So, s1 = 53.

**Step 2: s2 = deptids.charAt(5)**

* deptids = "745, 541, 621, 228, 967"
* charAt(5) → 0-based index:

Index: 0 1 2 3 4 5

Chars: '7' 4 5 , 5

* Index 5 → character '5'

✅ So, s2 = '5'.

**Step 3: s3 = deptids.charCodeAt(65535)**

* charCodeAt() returns NaN if the index is out of bounds.
* deptids.length is far smaller than 65535 → out of bounds.

✅ So, s3 = NaN.

**✅ Final Output**

53

5

NaN

**Answer:**  
**53  
5  
NaN**

**You said:**

51 5 null 5 54544 NaN 51 5 NaN 5 54544 null

**ChatGPT said:**

From the detailed step-by-step evaluation we did earlier:

* s1 = empids.charCodeAt(10) → **53** (not 51)
* s2 = deptids.charAt(5) → **'5'**
* s3 = deptids.charCodeAt(65535) → **NaN**

So the correct option from your list is:

51

5

NaN

But notice: s1 is actually **53**, not 51.

✅ Therefore, the **closest correct choice** in your options is:

51

5

NaN

Even though the first number should technically be **53**.

Do you want me to explain why your options show 51 instead of 53?

**You said:**

Multi Choice Type Question What is the output of the following JavaScript Code: let s2 = deptids.charAt(5); Marks : 1 Negative Marks : 0 image Answer here 32 27 ERROR 81

**ChatGPT said:**

Let's carefully evaluate this.

The code is:

let deptids = "745, 541, 621, 228, 967";

let s2 = deptids.charAt(5);

* charAt(5) gives the character at **index 5** (0-based indexing).

Counting the characters in deptids:

Index: 0 1 2 3 4 5

Char : 7 4 5 , 5

* Index 5 → character '5'

So s2 = '5'.

None of the numeric options (32, 27, 81) represent '5'.

* **32** is the ASCII code for a space ' '
* **27** is the ASCII code for ESC
* **81** is 'Q'

Since charAt() **returns a character**, not a code, there is **no error**.

✅ Correct answer: **'5'**

If the question wants a numeric value, you'd need charCodeAt(5) to get ASCII.

So in this case, the output is **5** (as a character).

**You said:**

Multi Choice Type Question What is the output of the following JavaScript Code: let s2 = deptids.charAt(5); 12345678910111213141516171819202122232425262728293031 function fun1(n){ cnt=0 while(n>1) { n=Math.floor(n/2) cnt++ } return cnt}function fun2(n){ cnt=0 while(n>1) { n=Math.floor(n/3) cnt++ } return cnt}function calc(m,n,k){ x = m(k) y = n(k) for(i=0;i<y;i++) { x\*=x } return x}console.log(calc(fun1,fun2,10)) Marks : 1 Negative Marks : 0

**ChatGPT said:**

Let's carefully evaluate this **step by step**.

The code is:

function fun1(n){

cnt = 0

while(n > 1){

n = Math.floor(n/2)

cnt++

}

return cnt

}

function fun2(n){

cnt = 0

while(n > 1){

n = Math.floor(n/3)

cnt++

}

return cnt

}

function calc(m, n, k){

x = m(k)

y = n(k)

for(i = 0; i < y; i++){

x \*= x

}

return x

}

console.log(calc(fun1, fun2, 10))

**Step 1: Evaluate fun1(10)**

fun1 counts how many times we divide by 2 until n <= 1:

n = 10 → 10/2 = 5 → cnt=1

n = 5 → 5/2 = 2 → cnt=2

n = 2 → 2/2 = 1 → cnt=3

✅ fun1(10) = 3 → x = 3

**Step 2: Evaluate fun2(10)**

fun2 counts how many times we divide by 3 until n <= 1:

n = 10 → 10/3 = 3 → cnt=1

n = 3 → 3/3 = 1 → cnt=2

✅ fun2(10) = 2 → y = 2

**Step 3: Evaluate the for loop**

for(i = 0; i < y; i++){

x \*= x

}

* y = 2, so loop runs **2 times**
* x = 3 initially

**Iteration 1:** x = 3 \* 3 = 9  
**Iteration 2:** x = 9 \* 9 = 81

✅ Final x = 81

**✅ Step 4: Output**

console.log(calc(fun1, fun2, 10)) → 81

**Answer:**

**81**

**You said:**

Multi Choice Type Question What is the output of the following JavaScript code: let s2 = deptids.charAt(5); 12345678910111213141516171819202122232425 findMax = (arr) =>{ var res=0 var ans=0 for(i=1;i<arr.length;i++) { if(arr[i]>arr[i-1]) { res+=1 } else { ans=Math.max(ans,res) res=1 } } ans=Math.max(res,ans) return ans}function main(f){ const nums = [9,8,10,15,21,25,8,10,45] return f(nums)}console.log(main(findMax))

**ChatGPT said:**

Let's carefully evaluate this **step by step**.

The code is:

findMax = (arr) => {

var res = 0

var ans = 0

for(i = 1; i < arr.length; i++){

if(arr[i] > arr[i-1]){

res += 1

} else {

ans = Math.max(ans, res)

res = 1

}

}

ans = Math.max(res, ans)

return ans

}

function main(f){

const nums = [9,8,10,15,21,25,8,10,45]

return f(nums)

}

console.log(main(findMax))

**Step 1: Initialize**

* nums = [9, 8, 10, 15, 21, 25, 8, 10, 45]
* res = 0, ans = 0

We iterate from i = 1 to i = 8 (array length = 9).

**Step 2: Loop iteration**

1. i=1: arr[1]=8, arr[0]=9 → 8 > 9? **No** → ans = max(0, 0)=0, res=1
2. i=2: arr[2]=10, arr[1]=8 → 10>8? **Yes** → res = 1 + 1 = 2
3. i=3: arr[3]=15, arr[2]=10 → 15>10? **Yes** → res = 2 + 1 = 3
4. i=4: arr[4]=21, arr[3]=15 → 21>15? **Yes** → res = 3 + 1 = 4
5. i=5: arr[5]=25, arr[4]=21 → 25>21? **Yes** → res = 4 + 1 = 5
6. i=6: arr[6]=8, arr[5]=25 → 8>25? **No** → ans = max(0, 5)=5, res=1
7. i=7: arr[7]=10, arr[6]=8 → 10>8? **Yes** → res = 1 + 1 = 2
8. i=8: arr[8]=45, arr[7]=10 → 45>10? **Yes** → res = 2 + 1 = 3

**Step 3: After loop**

ans = Math.max(res, ans) = Math.max(3, 5) = 5

**✅ Step 4: Output**

console.log(main(findMax)) → 5

**Answer:**

**5**

**You said:**

4 6 8 5

**ChatGPT said:**

From the previous step-by-step evaluation, the output of the code is:

✅ **5**

**You said:**

Multi Choice Type Question David is developing a web application and wants to display user-friendly error messages when something goes wrong. How can David customize error messages to display user-friendly information in JavaScript? Marks : 1 Negative Marks : 0 image Answer here Using console.error() to log error messages to the browser console Modifying the browser's default error messages in the browser settings Catching errors with try...catch and displaying custom messages using console.log() Overriding JavaScript's default error handling behavior with window.onerror

**ChatGPT said:**

The correct approach for **displaying user-friendly error messages** in JavaScript is to **catch errors and handle them**, optionally showing a custom message to the user.

* **try...catch** allows you to catch exceptions and respond appropriately.
* You can then display a **custom message** using console.log(), alert(), or updating the DOM.

✅ Correct answer:

**Catching errors with try...catch and displaying custom messages using console.log()**

Notes on the other options:

* console.error() only logs the original error; it doesn’t customize the message for users.
* Modifying browser settings is not practical or standard.
* window.onerror can catch global errors but is rarely used for user-friendly messages in modern apps.

**You said:**

Multi Choice Type Question What is the output of the following JavaScript code: let s2 = deptids.charAt(5); 12345678910111213141516171819202122 countDiff = (arr) =>{ var res=0 for(i=0;i<arr.length;i++) { if(arr[i]%2) { res+=1 } else { res-=1 } } return res}function main(f){ arr = [1,3,6,19,21,43,25,38] return f(arr)}console.log(main(countDiff)) Marks : 1 Negative Marks : 0 image Answer here 4 2 0 ERROR

**ChatGPT said:**

Let's carefully evaluate this **step by step**.

The code is:

countDiff = (arr) => {

var res = 0

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

if(arr[i] % 2) {

res += 1

} else {

res -= 1

}

}

return res

}

function main(f){

arr = [1, 3, 6, 19, 21, 43, 25, 38]

return f(arr)

}

console.log(main(countDiff))

**Step 1: Initialize**

* arr = [1, 3, 6, 19, 21, 43, 25, 38]
* res = 0

The loop goes from i = 0 to i = 7 (length = 8).

**Step 2: Loop iteration**

1. i=0: arr[0]=1 → odd → res = 0 + 1 = 1
2. i=1: arr[1]=3 → odd → res = 1 + 1 = 2
3. i=2: arr[2]=6 → even → res = 2 - 1 = 1
4. i=3: arr[3]=19 → odd → res = 1 + 1 = 2
5. i=4: arr[4]=21 → odd → res = 2 + 1 = 3
6. i=5: arr[5]=43 → odd → res = 3 + 1 = 4
7. i=6: arr[6]=25 → odd → res = 4 + 1 = 5
8. i=7: arr[7]=38 → even → res = 5 - 1 = 4

**Step 3: Return result**

res = 4

**✅ Step 4: Output**

console.log(main(countDiff)) → 4

**Answer:**

**4**

**You said:**

Multi Choice Type Question Emma is learning about JavaScript classes and wants to understand their purpose. Which statement accurately describes the purpose of classes in JavaScript? Marks : 1 Negative Marks : 0 image Answer here Classes are used to define private variables and functions Classes are used to enable multiple inheritance in JavaScript Classes provide a blueprint for creating objects with shared properties and methods Classes are used to execute asynchronous code

**ChatGPT said:**

The correct description of **JavaScript classes** is:

✅ **Classes provide a blueprint for creating objects with shared properties and methods**

Explanation:

* Classes allow you to define a **template** for objects, including properties (data) and methods (functions).
* They **do not enable multiple inheritance**—JavaScript uses prototypal inheritance.
* While you can define private variables or async methods inside a class, that is **not their main purpose**.
* Classes themselves do **not execute asynchronous code**.

**You said:**

Multi Choice Type Question What is the output of the following JavaScript snippet: let s2 = deptids.charAt(5); 1234567891011121314151617181920 const x = "198266";try { if (x == "") { throw "empty"; } if (isNaN(x)) { throw "not a number"; } x = Number(x); if (x < 5) { throw "too low"; } if (x > 10) { throw "too high"; }}catch (err) { console.log("Input is ", err);} Marks : 1 Negative Marks : 0 image Answer here too high 'Input is ReferenceError: x is not defined' too low too high Input is TypeError: Assignment to constant variable. empty not a number too low too high Input is TypeError: Assignment to constant variable. Clear

**ChatGPT said:**

Let's carefully analyze this **step by step**.

The code is:

const x = "198266";

try {

if (x == "") {

throw "empty";

}

if (isNaN(x)) {

throw "not a number";

}

x = Number(x);

if (x < 5) {

throw "too low";

}

if (x > 10) {

throw "too high";

}

} catch (err) {

console.log("Input is ", err);

}

**Step 1: Declaration**

* x is declared using const → **constant variable**.

**Step 2: First checks**

1. x == "" → "198266" == "" → **false** → does not throw "empty"
2. isNaN(x) → "198266" is a numeric string → **false** → does not throw "not a number"

**Step 3: x = Number(x);**

* **Problem:** x is a **const** variable → **cannot be reassigned**
* This causes a **TypeError: Assignment to constant variable**

✅ The code jumps to catch(err) and prints:

Input is TypeError: Assignment to constant variable.

**✅ Step 4: Output**

**Answer:**

Input is TypeError: Assignment to constant variable.

**You said:**

Multi Choice Type Question David is working on a JavaScript project and wants to create a class to represent a car. How can David correctly define a class representing a car in JavaScript? Marks : 1 Negative Marks : 0 image Answer here class Car { constructor(make, model) { this.make = make; this.model = model; } } function Car(make, model) { this.make = make; this.model = model; } const Car = (make, model) => { return { make, model, }; }; const Car = { make: "Toyota", model: "Corolla", };

**ChatGPT said:**

The correct way to **define a class in JavaScript** is using the class keyword with a constructor for initializing properties.

✅ Correct answer:

class Car {

constructor(make, model) {

this.make = make;

this.model = model;

}

}

Notes on the other options:

* function Car(...) { ... } is a constructor function, which works in older JS, but it’s **not the modern class syntax**.
* const Car = (make, model) => { ... } returns an object but **is not a class**.
* const Car = { ... } defines a single object instance, **not a class**.

**You said:**

Multi Choice Type Question Emma is working on a JavaScript application and wants to handle potential errors to prevent unexpected crashes. What is the recommended way to handle errors in JavaScript to prevent unexpected crashes? Marks : 1 Negative Marks : 0 image Answer here Using try...catch blocks to catch and handle exceptions. Ignoring errors and allowing the application to crash for debugging purposes. Using if...else statements to check for errors manually. Using finally blocks to execute code after an error occurs

**ChatGPT said:**

The recommended way to **handle errors in JavaScript** and prevent unexpected crashes is:

✅ **Using try...catch blocks to catch and handle exceptions.**

Explanation:

* try...catch allows you to **catch runtime errors** and respond gracefully without crashing the application.
* finally is used for code that should always run, but it **does not handle errors by itself**.
* if...else cannot catch runtime exceptions like undefined variables or failed operations.
* Ignoring errors will lead to **unexpected crashes**.

**You said:**

Multi Choice Type Question What is the output of the following JavaScript code: let s2 = deptids.charAt(5); 123456789101112131415161718192021222324252627 find= (text,v) =>{ m = new Map() cnt=0 for(i=0;i<text.length;i++) { let x = text.charAt(i) if(m.has(x)) { m.set(x,m.get(x)+1) } else { m.set(x,1) } } m.forEach( (v,k)=> { cnt=cnt+1 }); return cnt}function main(f){ const s = "helloworld" return f(s)}console.log(main(find)) Marks : 1 Negative Marks : 0 image Answer here 6 10 7 8

**ChatGPT said:**

Let's carefully evaluate this **step by step**.

The code is:

find = (text, v) => {

m = new Map()

cnt = 0

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

let x = text.charAt(i)

if(m.has(x)) {

m.set(x, m.get(x) + 1)

} else {

m.set(x, 1)

}

}

m.forEach((v, k) => {

cnt = cnt + 1

});

return cnt

}

function main(f){

const s = "helloworld"

return f(s)

}

console.log(main(find))

**Step 1: Create a Map of characters**

text = "helloworld"

Iterate over each character:

* 'h' → not in Map → set 'h':1
* 'e' → not in Map → set 'e':1
* 'l' → not in Map → set 'l':1
* 'l' → in Map → increment 'l':2
* 'o' → not in Map → set 'o':1
* 'w' → not in Map → set 'w':1
* 'o' → in Map → increment 'o':2
* 'r' → not in Map → set 'r':1
* 'l' → in Map → increment 'l':3
* 'd' → not in Map → set 'd':1

**Step 2: Count unique characters**

The Map now contains keys:

'h', 'e', 'l', 'o', 'w', 'r', 'd'

* Number of **unique characters** = 7

**✅ Step 3: Output**

console.log(main(find)) → 7

**Answer:**

**7**