



# CODING ASSIGNMENTS

Coding assignments that asses your  
Front-End skills



DEVELOPER UPDATES

- **Implement infinite scrolling**

Infinite scrolling is a popular technique for displaying content on a webpage. The user does not have to click on the next page of content but can scroll down to see more content.

**Technologies and concepts required:** HTML, CSS JavaScript, APIs, Async, Await in JavaScript, Fetch API, and Event listeners

**Example:** <https://www.javascripttutorial.net/javascript-dom/javascript-infinite-scroll/>

- **Design a Responsive Navigation Bar**

A responsive navigation bar is a website navigation menu that changes its size and shape depending on the screen size of the device used to view it. It also changes its position depending on the screen size as well.

**Technologies and concepts required:** HTML, CSS, CSS Media Queries, CSS Selectors

**Example:** [https://codepen.io/jo\\_Geek/pen/xgbaEr](https://codepen.io/jo_Geek/pen/xgbaEr)

- **Create a responsive grid layout**

A responsive grid layout is a type of design that displays content in a grid format, which can be rearranged to fit any screen size.

This type of design is beneficial for websites that are viewed on multiple devices and screens.

It eliminates the need to create different versions of the same website for each device and allows you to use one design template, which adjusts itself to fit any screen size.

**Technologies and concepts required:** HTML, CSS, CSS Media Queries, What is Grid Layout

**Example:** <https://travishorn.com/responsive-grid-in-2-minutes-with-css-grid-layout-4842a41420fe>

- **How do you validate form data?**

Form validation is a way to ensure that the form's user input is in the correct format.

**Technologies and concepts required:** HTML, JavaScript, regular expression

**Example:** <https://www.geeksforgeeks.org/form-validation-using-html-javascript/>

- **Make a To-do List App**

This is the most common question/assignment given by the interviewer to check the knowledge of the developer.

In the todo app, you will have to create a textbox with an add button and after adding a task it should be shown in the list below the textbox.

You can add advanced functionality like input validation, add, edit and delete task functionality with good design.

**Technologies and concepts required:** HTML, CSS, JavaScript, DOM Manipulation, Event listeners

**Example:** <https://www.tutorialstonight.com/to-do-list-javascript>

- **Build a calculator**

Create the layout of the calculator which is divided into two sections.

The first section is for the calculation buttons and the second section is for displaying the result of the calculation. And basic calculation buttons like +, -, /, \*, %, etc.

**Technologies and concepts required:** HTML, CSS, JavaScript, DOM Manipulation, Event listeners

**Example:** <https://www.makeuseof.com/build-a-simple-calculator-using-html-css-javascript/>

- **Implement EventEmitter in JavaScript**

Event emitters are a mechanism for one object to send a message to another object.

The event emitter pattern is used in JavaScript programming to implement publish/subscribe functionality.

An event emitter is an object that can emit events, which are objects that typically contain data about the event and at least one callback function to be executed when the event occurs.

**Example:** <https://css-tricks.com/understanding-event-emitters/>

- **Build a tic-tac-toe game**

The game is played on a 3×3 grid with alternating horizontal and vertical lines.

Players take turns placing their pieces on the board, with the player who places three pieces in a row, either horizontally, vertically, or diagonally being the winner.

**Technologies and concepts required:** HTML, CSS, JavaScript, Grid Layout, DOM Manipulation, Event listeners

**Example:** <https://css-tricks.com/understanding-event-emitters/>

- **Design a Developer Portfolio Web Page**

A developer portfolio is a web page that showcases the skills and experience of a developer.

**Technologies and concepts required:** HTML and CSS

**Example:** <https://codepen.io/ajibola-adeleke/pen/oNjmwOM>

- **Design a chatbox**

A chatbox is a small window on the side of a webpage, typically used for instant messaging.

The chatbox can be styled in any way you want, but it should provide the user with a place to type their message and send it to the person they are chatting with.

**Technologies and concepts required:** HTML and CSS

**Example:** <https://codepen.io/kristinak/pen/bXqayB>

- **Create a Photo Gallery**

Design the layout and appearance of the photo gallery using HTML and CSS. This involves creating a grid of photos using a flexbox or grid layout and styling the appearance of the photos, captions, and other elements. You can add more interactivity using JavaScript.

**Technologies and concepts required:** HTML, CSS, and JavaScript(Optional)

**Example:**

[https://www.w3schools.com/css/css\\_image\\_gallery.asp](https://www.w3schools.com/css/css_image_gallery.asp)

- **Implement a design provided in a mockup or wireframe.**

In this assignment, they will provide you with a mockup or wireframe. You will have to create a design that matches the provided mockup or wireframe.

**Technologies required:** They will tell you to use technologies that are given in the job description.

**Mockup examples:**

1. <https://dribbble.com/shots/20254545-Job-Portal-Website>
2. <https://dribbble.com/shots/19777764-Survey-Up-User-surveys>
3. <https://dribbble.com/shots/19111584-Pricing-page-Untitled-UI>
4. <https://dribbble.com/shots/18770188-OtFit-Website-Detail-Page>
5. <https://dribbble.com/shots/18436371-Subscription-Based-Design-Community>

- **Create a Slideshow / Carousel.**

A slideshow or rotating banner is a type of user interface that displays a series of images or other content in a cyclical fashion.

**Technologies and concepts required:** HTML, CSS, and JavaScript(Optional)

**Example:**

1. <https://css-tricks.com/css-only-carousel/>
2. <https://blog.logrocket.com/build-image-carousel-from-scratch-vanilla-javascript/>

- **Remove duplicate elements from an Array**

To remove duplicate elements from an array using JavaScript, you can use the following approach:

```
code.js

const array = [1, 4, 5, "Banana", 6, 1, 4, 4, 9, 3, "Banana"];

const uniqueElements = array.filter((element, index) => {
  return array.indexOf(element) === index
});

console.log(uniqueElements);

//Output: [1, 4, 5, "Banana", 6, 9, 3]
```

- **Write a Function to generate a Fibonacci series of a given length.**

```
code.js

function fibonacci(length) {
  // Initializing the array with the first two numbers of the series
  const fibArray = [0, 1];

  // Generating the rest of the series
  for (let i = 2; i < length; i++) {
    fibArray.push(fibArray[i-1] + fibArray[i-2]);
  }

  return fibArray;
}

const series = fibonacci(7);

console.log(series); //Output: [0, 1, 1, 2, 3, 5, 8]
```

- Reverse String using inbuilt methods and without methods

## Reversing string with methods

Converting string to array of characters

```
1 let string = "javascript";
2
3 let reversedString = string.split('').reverse().join('');
4
5 console.log(reversedString);
6 //Output: tpircsavaj
```

Reversing array elements

Getting array as a string

## Reversing string without methods

iterating over the string from the last character to the first character

```
1 let string = "javascript";
2
3 let reversedString = '';
4
5 for (let index = string.length - 1; index >= 0; index--) {
6   reversedString += string[index];
7 }
8
9 console.log(reversedString);
10 // Output: tpircsavaj
```

Decreasing index until the first character comes.

Appending characters



- Sort number array in descending order

```
1  const numbers = [23,34,123,43,5,77,38,2];  
2  
3  console.log(numbers.sort((num1, num2) => num2 - num1));  
4  
5  //Output: [123, 77, 43, 38, 34, 23, 5, 2]
```

array sort method

Compare function to sort an array

Result of num2-num1	Sorting Order
> 0	num2, num1
< 0	num1, num2
= 0	num2, num1

### Reference for compare function result

The sort method sorts data in ascending order according to Unicode value.

If you want to customize the sorting order, you will need to use a compare function, as shown in the example.

- Find the minimum and maximum value in the given array with inbuilt method and without inbuilt method.

## 1. Finding min and max using methods

```
1  const numbers = [23,34,123,43,5,77,38,2];  
2  
3  const min = Math.min(...numbers);  
4  console.log(min);  
5  //Output: 2  
6  
7  const max = Math.max(...numbers);  
8  console.log(max);  
9  //Output: 123
```

Method to find min value

Spread operator to get all numbers

Method to find max value

## 2. Finding min and max without using methods

Initialising min and max with first element of an array to compare with other elements

```
1  const numbers = [23,34,123,43,5,77,38,2];
2
3  var min = numbers[0];
4  var max = numbers[0];
5
6  for(var i = 1; i < numbers.length; i++) {
7      if(numbers[i] < min) {
8          min = numbers[i];
9      }
10     if(numbers[i] > max) {
11         max = numbers[i];
12     }
13 }
14
15 console.log(min);
16 //Output: 2
17 console.log(max);
18 //Output: 123
```

Looping over all numbers to compare

If current number is less than prev min replacing the min

If current number is greater than prev max replacing the min

- **Write a Linear Search Program.**

Linear search(sequential search), is a method for finding an element within a list or an array.

Linear search iterates through each element in an array, starting from the 0th index and searching until the element is found or the end of the array is reached.

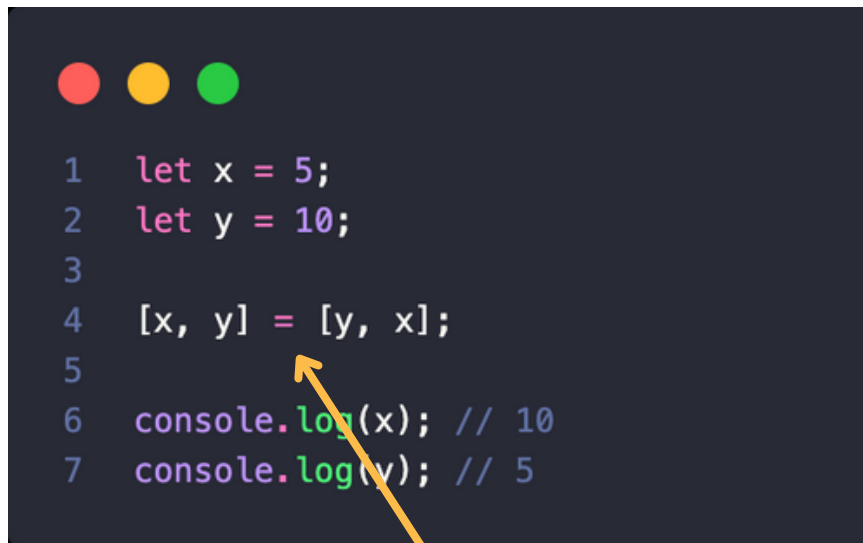
The image shows a code editor with a dark background and three colored window control buttons (red, yellow, green) at the top left. The code is as follows:

```
1  function linearSearch(array, searchValue) {
2
3      for (let i = 0; i < array.length; i++) {
4
5          if (array[i] === searchValue) {
6
7              return i;
8          }
9      }
10 }
11
12 return -1;
13 }
14
15 let numberArray = [1, 2, 3, 4, 5, 6];
16 let searchValue = 4;
17
18 console.log(linearSearch(numberArray, searchValue));
19
20 //Output: 3(Index of 4 is 3)
```

Three yellow callout boxes with arrows point to specific parts of the code:

- Loop to iterate an array**: Points to the `for` loop on line 3.
- If search value is found return index**: Points to the `return i;` statement on line 7.
- If element is not found**: Points to the `return -1;` statement on line 12.

- Swap the values of two variables without the need for a temporary third variable.



```
1  let x = 5;
2  let y = 10;
3
4  [x, y] = [y, x];
5
6  console.log(x); // 10
7  console.log(y); // 5
```

Destructuring assignment to swap the values by creating a new array

- Write a binary search program.

Binary search is an efficient search algorithm that allows for the searching of an ordered list of items.

It works by repeatedly dividing the list in half and searching for the target item in each half.

The algorithm starts by looking at the middle of the list and determines if the target item is in that position.

If it is not, it will then determine if the target item is in either the left or right half of the list and repeat this process until it finds the target item or determines that it is not present in the list.

Let's check the program:

Repeat the code till left meets right

```

1  function binarySearch(arr, searchVal) {
2
3      let left = 0; // start of the array
4      let right = arr.length - 1; //end of the array
5
6      while (left <= right) {
7
8          let mid = Math.floor((left + right) / 2);
9
10         if (arr[mid] === searchVal) {
11             return mid;
12         } else if (arr[mid] < searchVal) {
13             left = mid + 1;
14         } else {
15             right = mid - 1;
16         }
17     }
18
19     return -1;
20 }
21
22 let arr = [13, 21, 23, 34, 55, 86, 87, 98, 99];
23
24 let search = 55;
25
26 console.log(binarySearch(arr, search));
27
28 // Output: 4 (the index of 55 in the array)

```

Finding middle of the array

Return index if middle element is search Value

Element is at right part updating left index

Element is at left part updating right index

If element is not found

Binary search requires sorted array

- Write a program to print the following pattern.

```
*  
* *  
* * *  
* * * *  
* * * * *
```

Trick to understand the pattern(Use this trick to understand all pattern programs and write a program accordingly)

1. Find the number of rows

3. Find the number increment/decrement pattern(for number patterns(check the next example) as this is the plain \* pattern)

2. Find the number of columns

4. Write a program as follows

```

1  for(let i = 1; i <= 5; i++) {
2
3      let row = '';
4
5      for(let j = 1; j <= i; j++) {
6          row += '* ';
7      }
8  }

```

Loop for rows

New line after each row

Loop for Columns

Row values

- Write a program to print the following pattern.

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

```

1  for(let i = 1; i <= 5; i++) {
2
3      let row = '';
4
5      for(let j = 1; j <= i; j++) {
6          row += j + ' ';
7      }
8
9      console.log(row);
10 }

```

The logic will be same as above, here we are just changing number printing logic - j contains column value

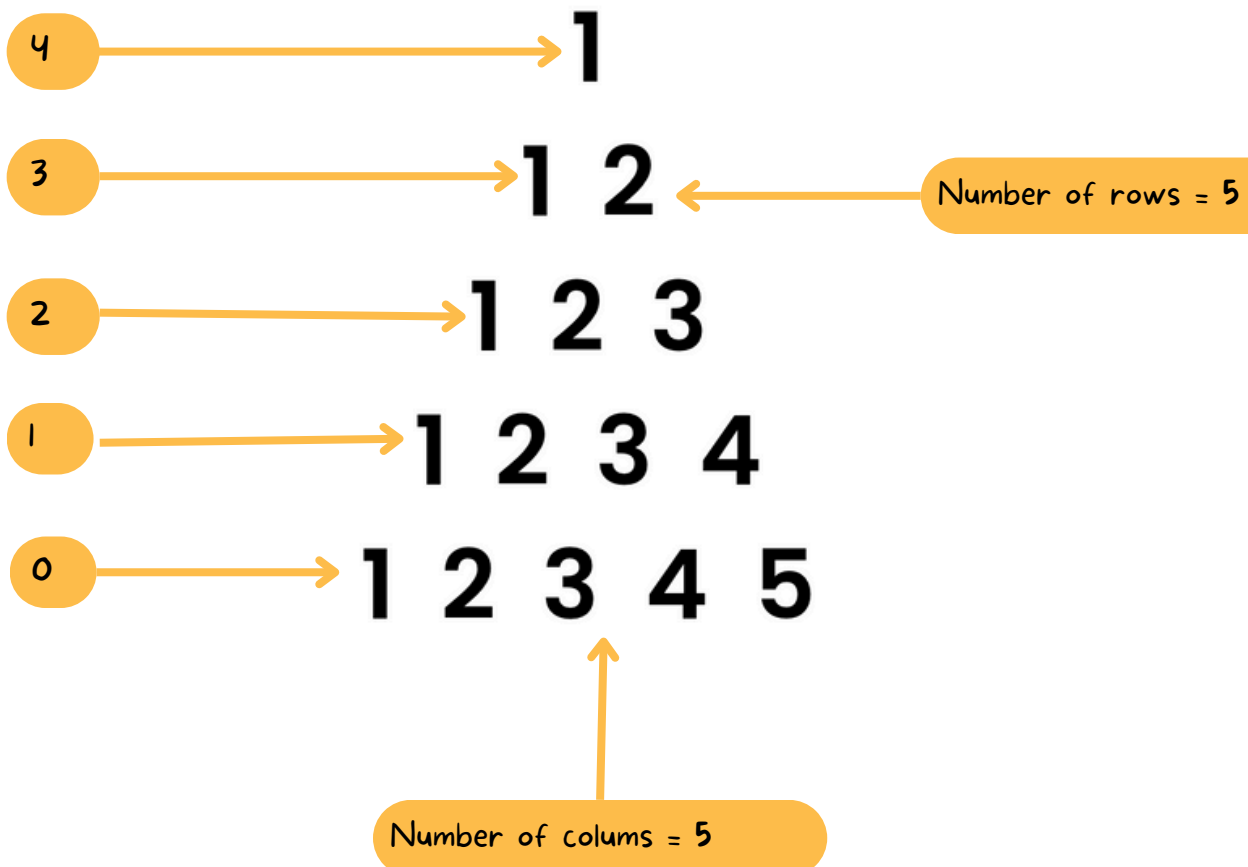


- Write a program to print the following pattern.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Let's first understand the pattern using above method:

Number of Spaces for each row (Total Number of rows - number of digits)



Now let's write a program for it:

```
1  function printPattern(rows)
2  {
3      for (let i = 1; i <= rows; i++) {
4          var output = '';
5
6          for (let j = 1; j <= rows - i; j++) {
7              output += ' ';
8          }
9
10         for (let k = 1; k <= i; k++) {
11             output += k + ' ';
12         }
13
14         console.log(output);
15     }
16 }
17
18 printPattern(5)
```

Loop to print rows

Loop to append spaces for each row

Loop to append Column Content(Digits)

Printing row content

- Write a program to print the following pattern.

```
  *
 * *
* * *
* * * *
* * * * *
```

The logic will be the same as the number pyramid(Above question), there will be only one change as below

```
1  function printPattern(rows)
2  {
3      for (let i = 1; i <= rows; i++) {
4          let output = '';
5
6          for (let j =1; j <= rows - i; j++) {
7              output += ' ';
8          }
9
10         for (let k = 1; k <= i; k++) {
11             output += '* ';
12         }
13
14         console.log(output);
15     }
16 }
17
18 printPattern(5)
```

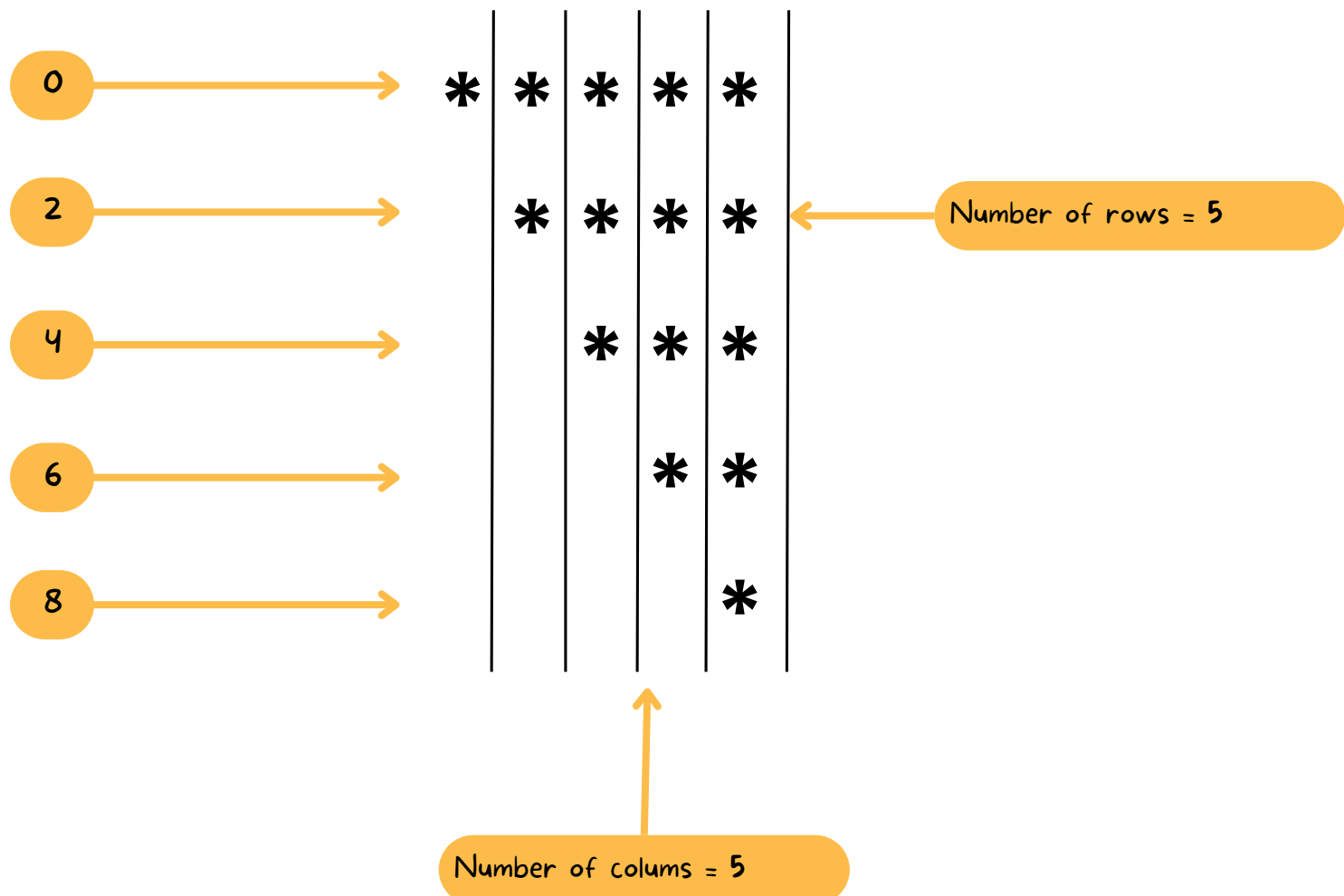
Append star instead of number

- Write a program to print the following pattern.

```
* * * * *
 * * * *
  * * *
   * *
    *
```

Lets first understand the pattern

Number of Spaces for each row (Total Number of rows - number of \* in row) \* 2 (blank space and space after star)



## Code to print the pattern

New Line

Loop to print rows

Loop to print Spaces

```
1  function printPattern(rows)
2  {
3
4    for (let i = rows; i >= 1; i--)
5    {
6
7      let row = "";
8
9      for (let j = 1; j <= (rows - i) * 2; j++)
10     {
11       row += " ";
12     }
13
14     for (let k = 1; k <= i; k++)
15     {
16       row += "* ";
17     }
18
19     console.log(row);
20   }
21 }
22
23 printPattern(5)
```

Loop to print \*

- Write a program to print the following pattern.

```
1 2 3 4 5
  1 2 3 4
    1 2 3
      1 2
        1
```

This pattern is same as above, you just have to print numbers instead of \*.

Code to print the pattern

```
1  function printPattern(rows)
2  {
3
4      for (let i = rows; i >= 1; i--)
5      {
6
7          let row = "";
8
9          for (let j = 1; j <= (rows - i) * 2; j++)
10         {
11             row += " ";
12         }
13
14         for (let k = 1; k <= i; k++)
15         {
16             row += k+" ";
17         }
18
19         console.log(row);
20     }
21 }
22
23 printPattern(5)
```

Append number instead of  
star

- Write a program to count the number of occurrences of a specific character in a string.

```
1  function countChars(str, char) {  
2    let count = 0;  
3  
4    const strLen = str.length;  
5  
6    for (let i = 0; i < strLen; i++) {  
7      if (str[i] === char) {  
8        count++;  
9      }  
10   }  
11   return count;  
12 }  
13  
14  
15 const string = "JavaScript is awesome!";  
16 const targetChar = "a";  
17  
18 const result = countChars(string, targetChar);  
19 console.log(result);  
20
```

Looping over all  
characters in a string

Comparing each  
character in a string

If character found,  
increasing count

- Write a program that removes duplicate characters from the string.

```
1 function removeDuplicateChars(str) {  
2  
3   const uniqueChars = new Set(str);  
4  
5   const result = Array.from(uniqueChars).join('');  
6  
7   return result;  
8 }  
9  
10 const inputString = "hello world";  
11 const result = removeDuplicateChars(inputString);  
12  
13 console.log("String without duplicate characters:", result);
```

Using "Set" to store unique data

Converting the set to an array, then joining the array to convert it into a string

- Write a program to print Chessboard using JavaScript

This question asks you to create a chessboard using JavaScript. A chessboard has 8 rows and 8 columns with black and white squares.

The interviewer wants to see if you can use loops to make this pattern and how you use JavaScript to show it on a webpage. It's a way to check your basic coding skills and how you solve problems.

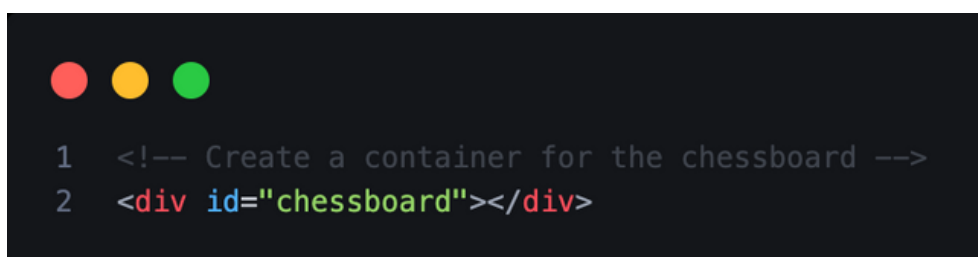
Here is the code to print the chessboard using JavaScript



- CSS



- HTML



- JavaScript

```
1  const chessboard = document.getElementById('chessboard');
2
3
4  for (let i = 0; i < 8; i++) {
5
6      for (let j = 0; j < 8; j++) {
7
8          const square = document.createElement('div');
9          square.classList.add('square');
10
11             if ((i + j) % 2 === 0) {
12                 square.classList.add('white');
13             } else {
14                 square.classList.add('black');
15             }
16
17             chessboard.appendChild(square);
18         }
19     }
```

Get chessboard container

Loop through each row

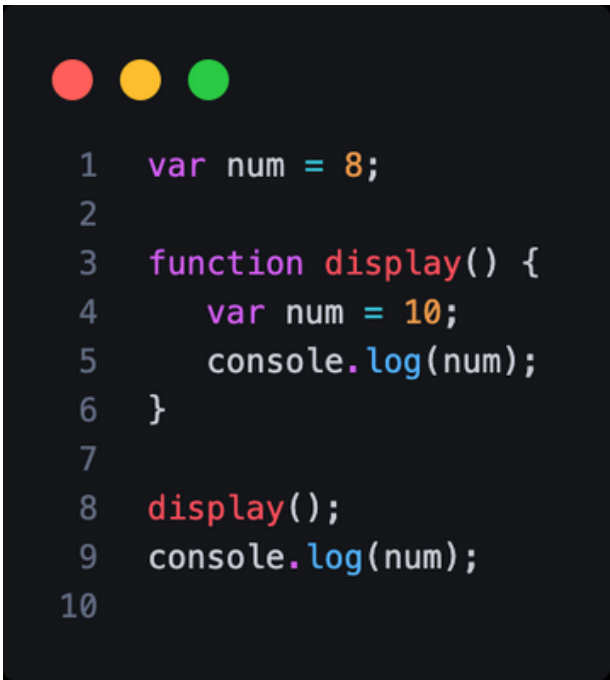
Loop through each Column

Create a new Square  
using Div

Set black and white color  
by even odd condition check

Add square to chessboard

- What will be the output of the following JavaScript code?



```
1  var num = 8;  
2  
3  function display() {  
4      var num = 10;  
5      console.log(num);  
6  }  
7  
8  display();  
9  console.log(num);  
10
```

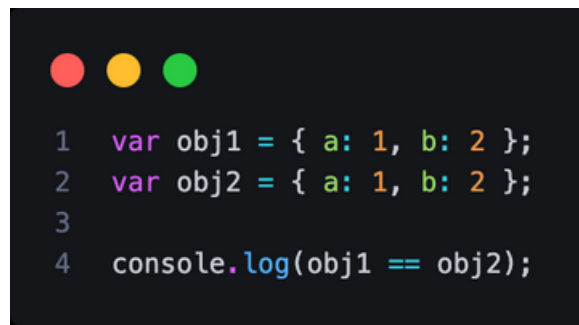
### Answer: 10 8

The code first defines a variable `num` with a value of 8. Then, a function `display` is defined which has its own local variable `num` with a value of 10.

Inside the `display` function, `console.log(num)` will log the local `num`, which is 10.

After the `display` function is called, the second `console.log(num)` logs the outer `num` (the global one), which is 8.

- What will the following code log to the console?



```
1 var obj1 = { a: 1, b: 2 };
2 var obj2 = { a: 1, b: 2 };
3
4 console.log(obj1 == obj2);
```

**Answer: false**

The comparison `obj1 == obj2` checks if both variables reference the same object in memory, not if their contents are the same.

Since `obj1` and `obj2` are two different objects in memory (despite having the same content), the result is `false`.

- Which method can be used to deeply clone an object in JavaScript?

- a) `Object.clone()`
- b) `Object.copy()`
- c) `JSON.parse(JSON.stringify(obj))`
- d) `Object.assign({}, obj)`

**Answer: c) `JSON.parse(JSON.stringify(obj))`**

- What does the following code return?

```
console.log("2" + 3 + 4;)
```

**Answer: "234"**

The code returns the string "234" due to string concatenation.

- **What will be the output of following code?**

```
for (var i = 0; i < 5; i++) {  
    setTimeout(function() { console.log(i); }, i * 1000 );  
}
```

**Answer: 5 5 5 5 5**

Think of `setTimeout` as setting a timer for each step in a task. In this code, you're setting up 5 timers very quickly, one after another.

However, because of how timers work, the task of setting them all up finishes first, before any of the timers go off.

Also, the code uses `var i` which means 'i' is shared across all these timers. When the loop finishes, 'i' becomes 5.

So, when the timers finally go off, they all see 'i' as 5 and therefore, each one prints 5.

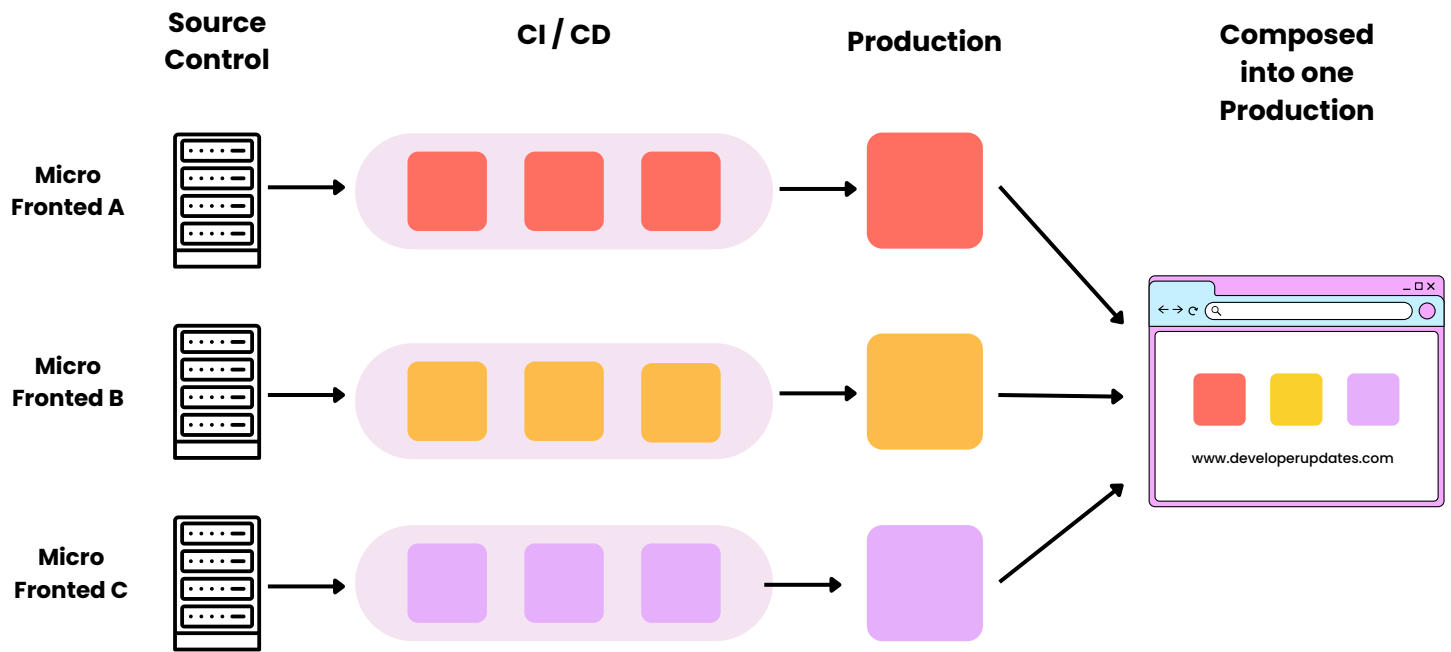
- **What is Micro Frontends?**

Micro Frontends is an architectural pattern for building web applications.

It involves breaking up a monolithic frontend application into smaller, semi-independent pieces called microfrontends.

Each microfrontend is a standalone web application, developed and deployed independently by different teams.

These microfrontends are then composed together into a single user interface.



## The key benefits of Micro Frontends are:

1. Improved team autonomy and scalability Teams can work independently on their own microfrontends without affecting others.
2. Technology freedom Different microfrontends can use different frameworks and technologies.
3. Simplified deployment and upgrades Changes to one microfrontend can be deployed without redeploying the entire application.
4. Improved fault isolation Issues in one microfrontend do not affect the entire application.

## **The composition of microfrontends can be achieved through various techniques like:**

- Iframes
- Web Components
- Server-Side Rendering
- Build-Time Integration

## **While Micro Frontends offer benefits, they also introduce complexities like:**

- Team coordination and communication overhead
- Potential performance issues due to multiple bundled assets
- Increased operational complexity

- **Traverse the DOM Tree**

To traverse the DOM (Document Object Model) level by level, we can use a breadth-first search (BFS) approach.

### **Here's how we can do it:**

1. Start with the root node (typically the `document.documentElement` or `document.body`).
2. Create a queue and enqueue the root node.
3. While the queue is not empty:
  - Dequeue a node from the front of the queue.
  - Process the dequeued node (e.g., log its value, modify it, etc.).

- Enqueue all the children of the dequeued node to the end of the queue

4. Repeat step 3 until the queue is empty.

This way, we visit all the nodes at the same level before moving to the next level.

We can implement this using a simple loop or a recursive function.

**Here's a JavaScript implementation using a loop:**



```
1  function traverseLevelByLevel(root) {
2      const queue = [root];
3
4      while (queue.length > 0) {
5          const levelSize = queue.length;
6
7          for (let i = 0; i < levelSize; i++) {
8              const node = queue.shift();
9              console.log(node.tagName); // Process the node
10
11              for (let j = 0; j < node.children.length; j++) {
12                  queue.push(node.children[j]);
13              }
14          }
15      }
16  }
17
18  traverseLevelByLevel(document.body);
```



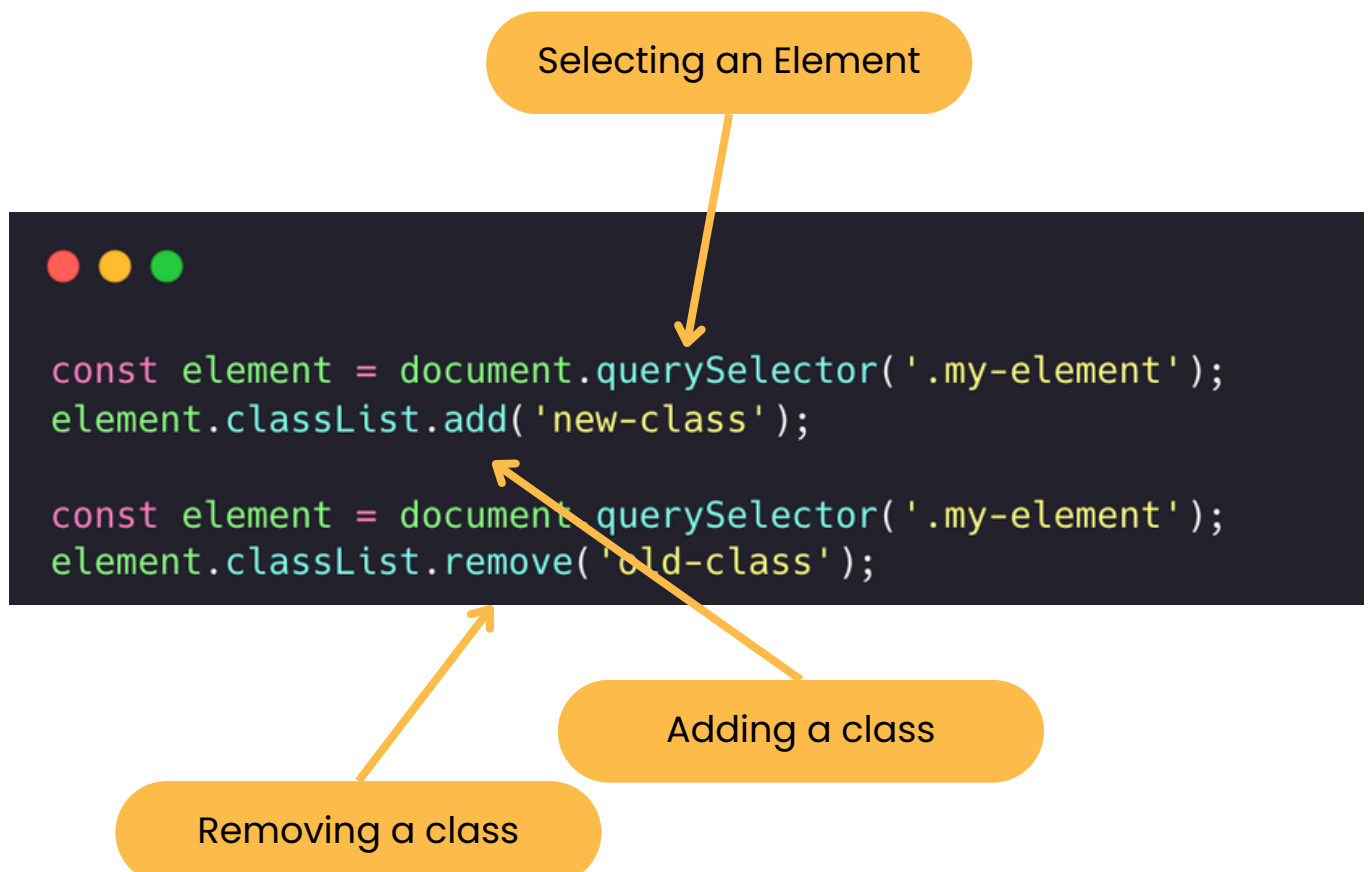
- **How do you optimize images for the web?**

Optimizing images for the web is crucial for improving website performance and providing a better user experience.

Here's how you can optimize images:

- Choose the right file format:
  - Use JPEG for photographs and complex images with many colors.
  - Use PNG for images with simple colors and transparency.
  - Use WebP or AVIF for better compression and quality.
- Resize images to the appropriate dimensions:
  - Resize images to the exact size they will be displayed on the website.
  - Larger images increase file size and slow down page load times.
- Compress images:
  - Use tools like TinyPNG, ImageOptim, or online compressors to reduce file size.
  - Lossless compression maintains image quality while reducing file size.
  - Lossy compression further reduces file size but may degrade image quality.
- Use responsive images:
  - Provide different image sizes for different screen resolutions and devices.
  - Use the srcset and sizes attributes in HTML or @media queries in CSS.
  -

- Leverage browser caching:
  - Set appropriate cache headers for images to enable browser caching.
  - Cached images don't need to be downloaded on subsequent page visits.
- Lazy load images:
  - Load images only when they are needed, as the user scrolls down the page.
  - This reduces initial page load time and improves perceived performance.
- Use Content Delivery Networks (CDNs):
  - Store and serve images from a globally distributed network of servers.
  - CDNs provide faster image delivery to users based on their location.
- **How do you add and remove a class from an element?**



- Write a program to print the Diamond pattern.

```
      *
     ***
    *****
   *******
  *********
 ***
**
*
```

### Step-by-Step Solution:

#### 1. Upper Part of the Diamond:

- Loop through each row from 1 to the specified number of rows.
- For each row, print leading spaces to center the asterisks.
- Print asterisks in a pattern that increases by 2 for each row (i.e., 1, 3, 5, ...).

#### 2. Lower Part of the Diamond:

- Loop through each row from the specified number of rows minus one down to 1.
- For each row, print leading spaces to center the asterisks.
- Print asterisks in a pattern that decreases by 2 for each row (i.e., ..., 5, 3, 1).

Here is the code to print diamond pattern:

```
function diamondPattern(rows) {  
  for (let row = 1; row <= rows; row++) {  
    let str = '';  
    for (let col = 1; col <= rows - row; col++) {  
      str += ' ';  
    }  
    for (let col = 1; col <= 2 * row - 1; col++) {  
      str += '*';  
    }  
    console.log(str);  
  }  
  
  for (let row = rows - 1; row >= 1; row--) {  
    let str = '';  
    for (let col = 1; col <= rows - row; col++) {  
      str += ' ';  
    }  
    for (let col = 1; col <= 2 * row - 1; col++) {  
      str += '*';  
    }  
    console.log(str);  
  }  
}  
  
diamondPattern(5);
```

Printing leading spaces

Upper Part

Printing Stars

Lower Part

Printing Stars

Printing leading spaces

- Write a program to print the Hollow Diamond pattern.

```
  *
 * *
*  *
*  *
*  *
*  *
*  *
*  *
*  *
 *
```

### Step-by-Step Solution:

#### 1. Upper Part of the Hollow Diamond:

- Loop through each row from 1 to the specified number of rows.
- For each row, print leading spaces to center the asterisks.
- Print asterisks at the beginning and end of the row, and fill the space between them with spaces.

#### 2. Lower Part of the Hollow Diamond:

- Loop through each row from the specified number of rows minus one down to 1.
- For each row, print leading spaces to center the asterisks.
- Print asterisks at the beginning and end of the row, and fill the space between them with spaces.

Here is the code to print diamond pattern:

Printing leading spaces

```

function hollowDiamondPattern(rows) {

  for (let row = 1; row <= rows; row++) {
    let str = '';

    for (let col = 1; col <= rows - row; col++) {
      str += ' ';
    }

    for (let col = 1; col <= 2 * row - 1; col++) {
      if (col === 1 || col === 2 * row - 1) {
        str += '*';
      } else {
        str += ' ';
      }
    }
    console.log(str);
  }

  for (let row = rows - 1; row >= 1; row--) {
    let str = '';

    for (let col = 1; col <= rows - row; col++) {
      str += ' ';
    }

    for (let col = 1; col <= 2 * row - 1; col++) {
      if (col === 1 || col === 2 * row - 1) {
        str += '*';
      } else {
        str += ' ';
      }
    }
    console.log(str);
  }
}

hollowDiamondPattern(5);

```

Upper Part

Printing Stars and Spaces

Lower Part

Printing Stars and Spaces

# Stay ahead with daily updates!

Follow us on social media for useful web development content.



[@richwebdeveloper](#)



[@new\\_javascript](#)



[@developerupdates](#)



[@developerupdates](#)



[@\\_chetanmahajan](#)

Note: This kit is continuously updated. Please continue to check Gumroad or our website (where you purchased this) for updated questions and answers. You will receive all updates for FREE

[Download from our Website](#)

[Download on Gumroad](#)

[WWW.DEVELOPERUPDATES.COM](http://WWW.DEVELOPERUPDATES.COM)