Questions and Answers JavaScript Evaluation Exam

Answer to Question 1 :

1. Html elements that aren't meant to store content or other elements are called ..... elements.

Void elements. Void elements in HTML are elements that do not have any content or closing tags. Examples include `<img>`, `<br>`, and `<input>`.

2. What are the differences between em, rem vs px?

- px: Stands for pixels. It's an absolute unit of measurement, meaning it specifies the exact number of pixels that an element will occupy. Using `px` ensures consistent sizing across different devices and resolutions.

- em: Relative unit of measurement. It is relative to the font-size of the element's parent. For example, if the parent element has a font-size of 16px, 1em will be equal to 16px. It scales relative to its container, making it useful for responsive design.

- rem: Stands for "root em". It is similar to `em` but relative to the font-size of the root element (`<html>`). For example, if the root element has a font-size of 16px, 1rem will always be equal to 16px, regardless of the font-size of its parent element. This provides a more predictable and consistent sizing compared to `em`.

3. What is the difference between `<input type="submit" value="click me">` and `<button type="submit">Click me</button>`?

- `<input type="submit" value="click me">`: This creates a submit button with the text "click me". It is an inline element and doesn't allow for any HTML content or child elements inside it. Customization is limited to CSS.

- `<button type="submit">Click me</button>`: This creates a submit button with the text "Click me". It is a block-level element and can contain HTML content and other elements such as images or additional text. It is more versatile and

easier to style and customize using CSS or adding other elements inside it.

4. Explain Hoisting and Event loop in JavaScript

- Hoisting: In JavaScript, hoisting is a behavior where variables and function declarations are moved to the top of their containing scope during the compilation phase. This means that you can use a variable or function before it is declared in the code. However, only the declarations are hoisted, not the initializations. For example:

```
console.log(x); // undefined
var x = 5;
console.log(x); // 5
```

In this case, the declaration of `x` is hoisted to the top, but the assignment `x = 5` is not.

- Event Loop: The event loop is a fundamental part of JavaScript's runtime environment that handles asynchronous operations. JavaScript is single-threaded, meaning it executes one operation at a time. The event loop continuously checks the call stack and the task queue. If the call stack is empty, it takes the first task from the queue and pushes it to the call stack for execution. This allows JavaScript to perform non-blocking operations, such as handling events, making network requests, or reading files, by deferring them to be executed later when the call stack is free.

5. What is the difference between for .. of, for .. in loop, and which JavaScript loop ensures that at least a singular iteration will happen?

- for..of: Iterates over iterable objects like arrays, strings, maps, sets, etc., returning the values of the iterable.

```
const array = [1, 2, 3];
for (const value of array) {
```

```
  console.log(value); // 1, 2, 3
}
```

- for..in: Iterates over enumerable properties of an object, returning the keys (property names).

```
const object = { a: 1, b: 2, c: 3 };
for (const key in object) {
  console.log(key); // a, b, c
}
```

- do..while: This loop ensures that at least one iteration will happen because the condition is evaluated after the first execution of the loop body.

```
let count = 0;
do {
  console.log(count); // Will always execute at least once
  count++;
} while (count < 5);
```

Answer to Question 2:

a.

Questions and Answers JavaScript Evaluation Exam

```
let c = { greeting: 'Hey!' };

let d;

d = c;

c.greeting = 'Hello';

console.log(d.greeting);

let x = 50;

let y = x;

x = 100;

console.log(y);
```

Answer:

The output is:

Hello

50

b.

```
var a = ['dog', 'cat', 'hen'];

a[100] = 'fox';

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

Answer:

The output is:

101

c.

```
const shape = {
  radius: 10,
  diameter() {
    return this.radius * 2;
  },
  perimeter: () => 2 * Math.PI * this.radius,
};
console.log(shape.diameter());
console.log(shape.perimeter());
```

Answer:

The output is:

20

NaN

d.

```
sum(10, 20);
diff(10, 20);
function sum(x, y) {
  return x + y;
}
let diff = function (x, y) {
```

```
  return x - y;
};
```

Answer:

The output is:

30

Uncaught ReferenceError: Cannot access 'diff' before initialization

e.

```
function myFunc(...args) {
  console.log(typeof(args))
}
myFunc()

function addNumbers(x, y, z) {
  const sum = x + y + z
  console.log(typeof(sum))
}
addNumbers(2, 10)
```

Answer:

The output is:

object

number

Questions and Answers JavaScript Evaluation Exam

NaN

Answer to question 3 :

1.

```
<section>
  <p>paragraph 1</p>
  <h2>Heading</h2>
  <p>paragraph 2</p>
  <p>paragraph 3</p>
</section>
```

```
h2 ~ p {
    color: blue;
  }
```

```
  h2 + p {
    background: beige;
  }
```

Answer:

The color: blue; style will be applied to:

paragraph 2

Questions and Answers JavaScript Evaluation Exam

paragraph 3

The background: beige; style will be applied to:

paragraph 2

2.

the best is the best way to mark up this layout

Answer:

```
<body>
    <div class="container">
        <h1>Mailing Address</h1>
        <address>
            6410 Via Real<br>
            Carpinteria, CA 93013<br>
            <a href="mailto:info@linkedin.com">info@linkedin.com</a>
        </address>
    </div>
</body>
```

3.

a function areAnagrams that checks if two strings are

anagrams of each other (they have the same characters but in

different orders).

console.log(areAnagrams('listen', 'silent')); // Output: true

Answer:

```
function areAnagrams(str1, str2) {
    // If the lengths of the strings are not the same, they cannot be anagrams
    if (str1.length !== str2.length) {
        return false;
    }

    // Sort the characters in the strings and compare
    return str1.split('').sort().join('') === str2.split('').sort().join('');
}
console.log(areAnagrams('listen', 'silent')); // Output: true
```

4.

a function `calculateMean` that calculates the mean (average) of all the numbers in an array.

Provide an example usage that outputs the mean of the array `[1, 2, 3, 4, 5]`.

Answer:

```
function calculateMean(numbers) {
    // Check if the array is empty
    if (numbers.length === 0) {
        return 0;
    }
    // Calculate the sum of all numbers in the array
    const sum = numbers.reduce((acc, num) => acc + num, 0);
    // Calculate the mean by dividing the sum by the number of elements
    return sum / numbers.length;
```

```
}

console.log(calculateMean([1, 2, 3, 4, 5])); // Output: 3
```

5.

a function `getSeason` that takes a month as input and returns the season it belongs to. The mapping of months to seasons is as follows:

- September, October, November -> Autumn
- December, January, February -> Winter
- March, April, May -> Spring
- June, July, August -> Summer

Provide an example usage that determines the season for the input month "October".

Answer:

```
function getSeason(month) {
    const autumn = ['September', 'October', 'November'];
    const winter = ['December', 'January', 'February'];
    const spring = ['March', 'April', 'May'];
    const summer = ['June', 'July', 'August'];

    if (autumn.includes(month)) {
        return 'Autumn';
    } else if (winter.includes(month)) {
        return 'Winter';
    } else if (spring.includes(month)) {
        return 'Spring';
```

```
    } else if (summer.includes(month)) {
        return 'Summer';
    } else {
        return 'Invalid month';
    }
}

// Example usage
console.log(getSeason('October')); // Output: Autumn
```

6.

a function in JavaScript to calculate the factorial of a number using recursion? A factorial is a mathematical operation represented by `n!`, which is the product of all positive integers less than or equal to `n`. For example, `factorial(4)` should return `24` (since 4 * 3 * 2 * 1 = 24).

Answer:

```
function factorial(n) {
    // Base case: if n is 0 or 1, the factorial is 1
    if (n === 0 || n === 1) {
        return 1;
    }
    // Recursive case: multiply n by the factorial of n-1
    return n * factorial(n - 1);
}
```

```
// Example usage
console.log(factorial(4)); // Output: 24
console.log(factorial(5)); // Output: 120
```

7.

a function `duplicate` in JavaScript that takes an array as input and returns a new array where the original array is duplicated. For example, `duplicate([1, 2, 3, 4])` should return `[1, 2, 3, 4, 1, 2, 3, 4]`.

Answer:

```
function duplicate(arr) {
    return arr.concat(arr);
}

// Example usage
console.log(duplicate([1, 2, 3, 4])); // Output: [1, 2, 3, 4, 1, 2, 3, 4]
```

8.

How can you empty an array in JavaScript using four different methods?

Answer:

1. Setting the length to 0:

```
    let arr = [1, 2, 3, 4];
    arr.length = 0;
    console.log(arr); // Output: []
```

Explanation: Setting the length property of the array to 0 removes all elements from the array.

2. Assigning a new empty array:

```
let arr = [1, 2, 3, 4];
arr = [];
console.log(arr); // Output: []
```

Explanation: Assigning a new empty array to the variable effectively removes all references to the old array, making it empty.

3. Using the `splice` method:

```
let arr = [1, 2, 3, 4];
arr.splice(0, arr.length);
console.log(arr); // Output: []
```

Explanation: The `splice` method modifies the array in place by removing elements from the start index (0) to the end index (`arr.length`).

4. Using a while loop:

```
let arr = [1, 2, 3, 4];
while (arr.length > 0) {
    arr.pop();
}
console.log(arr); // Output: []
```

Explanation: The while loop continues to remove the last element of the array using `pop` until the array is empty.

Answer to question 4 :

1.

implement a feature in HTML, CSS, and JavaScript that allows a user to hide and show content by clicking on "Show" and "Hide" buttons on this Github link:

https://github.com/mahitab77/javascript-training/tree/main/js-ass-hide-show-content

2.

create a contact form layout as shown using only HTML and CSS

https://github.com/mahitab77/javascript-training/tree/main/contact-form-using%20-css-html(Answer%20question%204.2%20in%20evaluation%20exam)