Question 1-5 contains 1 mark each

1. Which of the following is the correct way to create a list using the lowercase letters?

a. <ol alpha=”a”> b. <ol type=”a”> c. <ol letter=”a”> d. None

Answer :

b. <ol type=”a”>

2. In HTML5, which of the following tag is used to initialize the document type?

a. <Doctype HTML> b. <\Doctype html> c. <DOCTYPE> d. <!DOCTYPE html

Answer :

d. <!DOCTYPE html>

3. Which of the following has the highest priority in CSS?

a. #id b. HTML tags c. .my-class d. :hover

Answer :

c. .my-class

4. The default value of “position” property is .

a. fixed b. absolute c. static d. relative

Answer :

c. static

5. How to stop event bubbling in JavaScript?

a. e.stopBubble() b. e.stop() c. e.pauseBubble() d. e.stopPropagation()

Answer :

d. e.stopPropagation()

6 . Explain Closure, Hoisting and Currying (Function Composition) in JavaScript, with the help of examples. (10 Marks)

Answer:

To elaborate, the parent function's variables are accessible to the inner function.

If the inner function uses its parent's (or parent's parent's and so on) variable(s)

then they will persist in memory as long as the accessing functions(s) are still referenceable.

In JavaScript, referenceable variables are not garbage collected.

Let's review the identity function:

function identity(a) { return a; }

Closures provide a way to associate data with a method that operates on that data.

They enable private variables in a global world.

Many patterns, including the fairly popular module pattern, rely on closures to work correctly.

Example :

```
function foo(x) {
  function bar(y) {
    console.log(x + y); }
  bar(2);
}
foo(2);
```

The outer function (foo) takes a variable (x), which, which is bound to that function when invoked. When the internal function (bar) is invoked, x (2) and y (2) are added together then logged to the console as 4. Bar is able to access foo's x-variable because bar is created within foo's scope.

Currying is the process of transforming a function with many arguments into the same function with less arguments.

That sounds cool, but why would I care about that?

Currying can help you make higher order factories.

Currying can help you avoid continuously passing the same variables.

Currying can memorize various things including state.

Let's pretend that we have a function (curry) defined and set onto the function prototype which turns a function into a curried version of itself. Please note, that this is not a built in feature of JavaScript.

```
function msg(msg1, msg2) {

  return msg1 + ' ' + msg2 + '.';

}

var hello = msg.curry('Hello,');

console.log(hello('Sarah Connor')); // Hello, Sarah Connor.

console.log(msg('Goodbye,', 'Sarah Connor'));
```

Hoisting.

Hoisting in JavaScript is a behavior in which a function or a variable can be used before declaration. For example,

```
console.log(test);   // undefined

var test;
```

Since the variable test is only declared and has no value, undefined value is assigned to it.

In terms of variables and constants, keyword var is hoisted and let and const does not allow hoisting.

For example,

```
// program to display value

a = 5;

console.log(a);

var a; // 5
```

In the above example, variable a is used before declaring it. And the program works and displays the output

7.Demonstrate using code sample, multiple ways of using CSS in HTML. What is the priority order of the same? (10 Marks)

Answer:

There are many ways to use the css. Css can be used inline, internal, external to the file.

Inline css : Usually, CSS is written in a separate CSS file (with file extension .css) or in a <style> tag inside of the <head> tag, but there is a third place which is also valid. The third place you can write CSS is inside of an HTML tag, using the style attribute

Example :

```
<html>
  <head>
    <title>Playing with Inline Styles</title>
  </head>
  <body>
    <p style="color:blue;font-size:46px;">
```

```
    I'm a big, blue, <strong>strong</strong> paragraph
  </p>
 </body>
</html>
```

The style attribute is just like any other HTML attribute. It goes inside the element's beginning tag, right after the tag name. The attribute starts with style, followed by an equals sign, =, and then finally uses double quotes, "", which contain the value of the attribute.

In order with the other way of using the css Inline css has the first priority to others ways.

Internal Css : Internal CSS in HTML means adding CSS code in the <head> section of the document. Styling changes apply to every specific element found in the file.

You can integrate internal CSS stylesheets by placing the <style> element in the <head> section of a page.

Internal styles apply to whole pages but not to multiple HTML documents.

Several pages can be styled by repeating the same block of internal styles in them.

```
<head>
  <style>
    h1 {
      color: red;
      margin-left: 20px;
    }
    p {
      color: blue;
```

```
    }
  </style>
</head>
```

Internal css has the second priority after Inline css.

External Css :

The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file.

It uses the <link> tag on every pages and the <link> tag should be put inside the head section.

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

The external style sheet may be written in any text editor but must be saved with a .css extension. This file should not contain HTML elements.

Let's take an example of a style sheet file named "mystyle.css".

File mystyle.css

```
body {
    background-color: lightblue;  }
h1 {
    color: navy;
    margin-left: 20px;
}
```

External css has the last priority after Inline and Internal css

8. Find and explain the output of the following code snippet

```
const arr = [10, 12, 15, 21];
for (var i = 0; i < arr.length; i++) {
  setTimeout(function() {
    console.log('Index: ' + i + ', element: ' + arr[i]);
  }, 3000);
}
```

Answer:

Index:0, element10

Index:1, element12

Index:2, element15

Index:3, element21

we have used setTimeout Function to Increase the output time of every console output. with 3000ms

9. Find and explain the output of the following code snippet.

```
const promise1 = new Promise((resolve, reject) => {
  console.log(1);
  resolve('success')
});
promise1.then(() => {
  console.log(3);
});
console.log(4);
```

Ans.

1

4

3

At stage one once new promise is on the way it will console 1 and will wait for the success and after success it will wait for the response to send back

and mean time it will console 4 and once response received at server end it will console 3