Q1. What is the difference between Cookie / Local Storage / Session Storage?

Ans: **Cookie:** When a user accesses a website, a web server generates small text files known as cookies, and the user then keeps these data on their device. Cookies are used to track user behavior, remember user preferences, and deliver customized web experiences. A key-value pair and an expiration date are both part of a cookie. The value, which contains details about the user or their activity on the website, can be retrieved using the key, a special identifier. Cookies have expiration dates before being uninstalled from a device automatically.

**Local Storage:** Web apps save data locally on the user's device using a browser feature called local storage. Local storage is usually in the form of key-value pairs, and it stores data with no expiration date and is stored in the browser's memory. The key is a unique identifier, associated with a specific value, and is use to retrieve the related value. This data can still be read and modified by the program even when the user closes the browser and opens it again later. Despite the similarities between local storage and cookies, there are a few significant variations.

**Session Storage:** Web developers can keep data on a user's device for a single browsing session. Session storage is only available till the session is present, this means it will erase all the data after user exits the website. We can say that it creates a temporary storage object which is scoped to the current browsing context and is not shared between different browser tabs or windows.

Q2. What is the significance of, and reason for, wrapping the entire content of a JavaScript source

file in a function block?

Ans: It creates a closure around the content of the file which basically creates a private namespace which helps in avoiding any name clashes between different javascript libraries and modules.or we can say that wrapping JavaScript code in a function block is done to for 2 main reasons:

1. Encapsulate variables and functions to avoid conflicts.
2. Prevent polluting the global scope.

Q3. What will the code below output? Explain your answer.

console.log(0.1 + 0.2);

console.log(0.1 + 0.2 == 0.3);

Ans: console.log(0.1 + 0.2): 0.3

console.log(0.1 + 0.2 == 0.3): false

Q4. Write a sum method which will work properly when invoked using either syntax below.

console.log(sum(2,3)); // Outputs 5

console.log(sum(2)(3)); // Outputs 5

Ans: function sum(a, b) {

if (arguments.length === 2) {

return a + b;

} else if (arguments.length === 1) {

return function(b) {

return a + b;

};

}

}

Q5. What is the output of the following code?

var length = 10;

function fn() {

console.log(this.length);

}

var obj = {

length: 5,

method: function(fn) {

fn();

arguments[0]();

}

};

obj.method(fn, 1);

Ans: 10

2