1. Write a function that creates a closure and returns a function that can add a specific number to any number passed to it. For example, if the closure is created with 5, the returned function should add 5 to any number passed to it.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Functions</h2>

<p>A function can access variables defined outside the function:</p>

<p id="demo"></p>

<script>

var a = 5;

myFunction();

function myFunction()

{

var num = prompt("Enter new number");

var n = (num + a).Number();

document.getElementById("demo").innerHTML = n;

}

</script>

</body>

</html>

1. Write a recursive function that searches an array for a specific value. The function should return true if the value is found, and false if it is not. You can assume that the array is not nested.

function searchArray(array, value) {

if (array.length === 0) {

return false;

}

if (array[0] === value) {

return true;

} else {

return searchArray(array.slice(1), value);

}

}

1. Write a function that adds a new paragraph element to the bottom of an HTML document. The function should take a string argument that will be used as the text content of the new paragraph element.

function addParagraph(text)

{

const newParagraph = document.createElement("p");

newParagraph.textContent = text;

document.body.appendChild(newParagraph);

}

1. Write a function that adds a new list item to an unordered list in an HTML document. The function should take a string argument that will be used as the text content of the new list item.

function addListItem(text)

{

const list = document.querySelector("ul");

const newListItem = document.createElement("li");

newListItem.textContent = text;

list.appendChild(newListItem);

}

1. Write a function that changes the background color of an HTML element. The function should take two arguments: the first argument is a reference to the HTML element, and the second argument is a string representing the new background color.

function changeBackgroundColor(element, color)

{

element.style.backgroundColor = color; // set the background color of the element

}

const myElement = document.getElementById("my-element");

changeBackgroundColor(myElement, "blue");

1. Write a function that saves an object to local Storage. The function should take two arguments: the first argument is a string representing the key to use for storing the object, and the second argument is the object to store.

function saveToLocalStorage(key, obj) {

localStorage.setItem(key, JSON.stringify(obj));

}

1. Write a function that retrieves an object from local Storage. The function should take one argument, which is a string representing the key used to store the object. The function should return the object

function getObjectFromLocalStorage(key) {

const item = localStorage.getItem(key);

if (!item) {

return null;

}

try {

return JSON.parse(item);

} catch (error) {

console.error(`Error parsing JSON from localStorage: ${error}`);

return null;

}

}

1. Write a function that takes an object and saves each property to local Storage using the property name as the key and the property value as the value. The function should also retrieve the object from local Storage and return it as a new object

function saveObjectToLocalStorage(obj) {

for (const prop in obj) {

localStorage.setItem(prop, JSON.stringify(obj[prop]));

}

const newObj = {};

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

const key = localStorage.key(i);

newObj[key] = JSON.parse(localStorage.getItem(key));

}

return newObj;

}