script.js

```javascript
// 1. Write a function called copyArray that takes in an array and returns a shallow copy of that array.
function copyArray(arr) {
  return [...arr];
}

console.log(copyArray([1, 2, 3])); // [1, 2, 3]
console.log(copyArray(["Bob", "Mike", "David"])); // ['Bob', 'Mike', 'David']

// 2. Given an object like the one below, write a function that takes an object as an argument and returns a nested array with the object's key/value pairs inside nested arrays. For example, an object like this { a: 1, b: 2 } should return [[a, 1], [b, 2]].
// const sampleFoodObj = {
//   fruit: "apples",
//   meat: "beef",
//   dairy: "milk",
//   dessert: "cake",
// };

function createArray(obj) {
  return Object.entries(obj);
}

console.log(createArray({ a: 1, b: 2 })); // [['a', 1], ['b', 2]]
console.log(createArray({ firstName: "Mike", lastName: "Smith", age: 27 })); // [['firstName', 'Mike'], ['lastName', 'Smith'], ['age', 27]]

// 3. Write a function called arraysEqual that takes two arrays and returns true if they contain the same elements in the same order.
function arraysEqual(arr1, arr2) {
  return arr1.join() === arr2.join();
}

console.log(arraysEqual([1, 2, 3], [1, 2, 3])); // true
console.log(arraysEqual([1, 2, 3], [4, 5, 6])); // false
console.log(arraysEqual([], [])); // true
console.log(arraysEqual(["Hello", "World"], ["Hello", "World"])); // true

// 4. Write a function called sortDescending that takes an array of numbers and returns a new array sorted from highest to lowest.

function sortDescending(arr) {
  return arr.sort((a, b) => b - a);
}

console.log(sortDescending([1, 2, 3])); // [3, 2, 1]
console.log(sortDescending([19, 11, 27])); //[27, 19, 11]

```

```javascript
// 5. Write a function called flattenOnce that takes a nested array (1-level deep) and returns a flat array
function flattenOnce(arr) {
  return arr.flat(1);
}

console.log(flattenOnce([1, [2, 3], 4])); // [1, 2, 3, 4]
console.log(flattenOnce(["Stan", 27, [{ a: 1, b: 2 }]])); // ["Stan", 27, {a: 1, b: 2}]

// 6. Write a function called getKeyValueArrays that takes an object and returns an object with two properties: one called keys containing the object’s keys, and one called values containing the object’s values.
function getKeyValueArrays(obj) {
  return { keys: Object.keys(obj), values: Object.values(obj) };
}

console.log(getKeyValueArrays({ name: "Leo", age: 30 })); // { keys: ["name", "age"], values: ["Leo", 30] }
console.log(getKeyValueArrays({ city: "Denver", high: 87, low: 66 })); // { keys: ["city", "high", "low"], values: ["Denver", 87, 66] }

```