# DSA QUESTIONS WITH & WITHOUT FUNCTION

Let’s look at array operations **with and without functions** in PHP and JavaScript. This will help clarify how to perform operations both directly (manual logic) and using built-in functions.

**1. Array Operations Without Functions**

**Traversal**

**PHP:**

$arr = [1, 2, 3, 4];

for ($i = 0; $i < count($arr); $i++) {

echo $arr[$i] . "\n";

}

**JavaScript:**

let arr = [1, 2, 3, 4];

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

console.log(arr[i]);

}

**Insertion**

**PHP:**

$arr = [1, 2, 4];

for ($i = count($arr); $i > 2; $i--) {

$arr[$i] = $arr[$i - 1];

}

$arr[2] = 3; // Insert 3 at index 2

print\_r($arr);

**JavaScript:**

let arr = [1, 2, 4];

for (let i = arr.length; i > 2; i--) {

arr[i] = arr[i - 1];

}

arr[2] = 3; // Insert 3 at index 2

console.log(arr);

**Deletion**

**PHP:**

$arr = [1, 2, 3, 4];

for ($i = 2; $i < count($arr) - 1; $i++) {

$arr[$i] = $arr[$i + 1];

}

unset($arr[count($arr) - 1]); // Remove the last element

print\_r($arr);

**JavaScript:**

let arr = [1, 2, 3, 4];

for (let i = 2; i < arr.length - 1; i++) {

arr[i] = arr[i + 1];

}

arr.length--; // Remove the last element

console.log(arr);

**Search**

**PHP:**

$arr = [1, 2, 3, 4];

$target = 3;

$found = false;

for ($i = 0; $i < count($arr); $i++) {

if ($arr[$i] === $target) {

echo "Element found at index $i\n";

$found = true;

break;

}

}

if (!$found) echo "Element not found.\n";

**JavaScript:**

let arr = [1, 2, 3, 4];

let target = 3;

let found = false;

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

if (arr[i] === target) {

console.log(`Element found at index ${i}`);

found = true;

break;

}

}

if (!found) console.log("Element not found.");

**2. Array Operations With Built-in Functions**

**Traversal**

**PHP:**

$arr = [1, 2, 3, 4];

array\_map(function($value) {

echo $value . "\n";

}, $arr);

**JavaScript:**

let arr = [1, 2, 3, 4];

arr.forEach(value => console.log(value));

**Insertion**

**PHP:**

$arr = [1, 2, 4];

array\_splice($arr, 2, 0, 3); // Insert 3 at index 2

print\_r($arr);

**JavaScript:**

let arr = [1, 2, 4];

arr.splice(2, 0, 3); // Insert 3 at index 2

console.log(arr);

**Deletion**

**PHP:**

$arr = [1, 2, 3, 4];

array\_splice($arr, 2, 1); // Remove element at index 2

print\_r($arr);

**JavaScript:**

let arr = [1, 2, 3, 4];

arr.splice(2, 1); // Remove element at index 2

console.log(arr);

**Search**

**PHP:**

$arr = [1, 2, 3, 4];

$target = 3;

$index = array\_search($target, $arr);

if ($index !== false) {

echo "Element found at index $index\n";

} else {

echo "Element not found.\n";

}

**JavaScript:**

let arr = [1, 2, 3, 4];

let index = arr.indexOf(3);

if (index !== -1) {

console.log(`Element found at index ${index}`);

} else {

console.log("Element not found.");

}

**3. Time and Space Complexity Comparison**

|  |  |  |
| --- | --- | --- |
| **Operation** | **Without Function** | **With Function** |
| **Traversal** | O(n) | O(n) |
| **Insertion** | O(n) (manual shift needed) | O(n) (handled internally) |
| **Deletion** | O(n) (manual shift needed) | O(n) (handled internally) |
| **Search** | O(n) | O(n) |

**When to Use Which Approach?**

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Without Functions** | **With Functions** |
| **Custom Behavior** | Use when needing full control of logic. | Not recommended for complex operations. |
| **Code Simplicity** | Can get verbose for complex scenarios. | Ideal for concise, readable code. |
| **Performance Optimization** | No internal overhead, faster in theory. | Efficient for standard operations. |

By combining manual logic with built-in functions, you can optimize operations depending on the problem’s requirements.