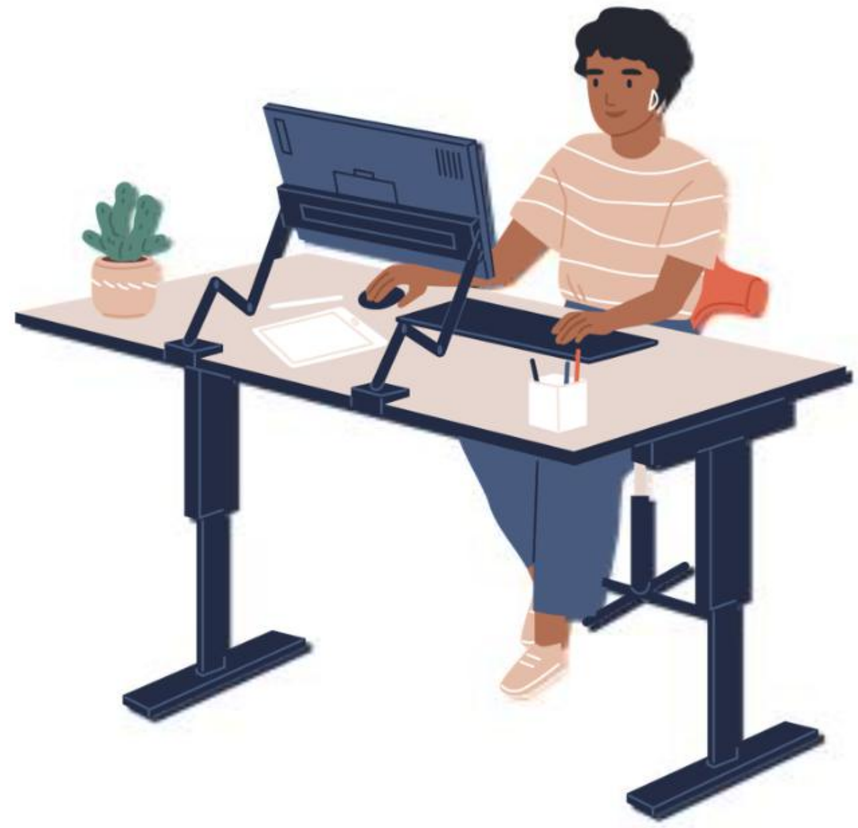
Slide Note

Learning Consolidation

Perform Common Array Operations Using JavaScript Array Methods



Slide Note

In this Sprint, you learned to:

- Demonstrate common array operations using array methods
- Explain function expressions and arrow functions
- Filter, transform and aggregate data using array methods
- Perform operations on complex data structures

Slide Note

Array Operations

Insertion	• push(), unshift()
Removal	• pop(), shift(), splice()
Traversal	• forEach()
Transformation	• map(), filter(), slice()
Search	• find(), indexOf()
Aggregation	• reduce()

Slide Note

```
let colors = ['Red','Blue'];
colors.push( 'Yellow', 'Green');
console.log(colors);
//['Red', 'Blue', 'Yellow', 'Green']

let removedElement = colors.pop();
console.log( removedElement);
//['Green']
console.log(colors);
//['Red', 'Blue', 'Yellow']
```

Built-in Methods: push and pop

- push(): Adds one or more elements to the end of an array and returns the array's new length.
- pop(): Removes the last element from an array and returns that element.
- Both methods modify the original array.

Slide Note

Built-in Methods: shift and unshift

- shift(): Removes the first element from an array and returns that element.
- unshift(): Adds new elements to the beginning of an array and returns the array's new length.
- Both methods modify the original array.

```
let cars=['BMW', 'Ford', 'Ferrari'];
let removedCar =  cars.shift();
Console.log(removedCar);
//'BMW'
console.log(cars);
// ['Ford', 'Ferrari']

cars.unshift('Land Rover');
Console.log(cars);
//['Land Rover', 'Ford', 'Ferrari']
```

Slide Note

```
const colors = [ 'Violet', 'Indigo',
'Blue', 'Green', 'Yellow', 'Orange',
'Red'];

colors.sort();
console.log(colors);
// ['Blue', 'Green', 'Indigo',
'Orange', 'Red', 'Violet', 'Yellow']

colors.reverse();
console.log(colors);
//['Red', 'Orange', 'Yellow','Green',
'Blue', 'Indigo', 'Violet']
```

Rearrange Elements in an Array

- sort(): Sorts the elements of an array. The default sort order is ascending.
- reverse(): Reverses the order of the elements in an array. The first array element becomes the last, and the last array element becomes the first.

Slide Note

Search Elements in an Array

- indexOf(): Returns the first index at which a given element can be found in the array, or -1 if it is not present.
  - The array is searched starting at the index position specified as the second parameter.
- lastIndexOf(): Returns the last index at which a given element can be found in the array, or -1 if it is not present.
  - The array is searched backward starting at the index position specified as the second parameter.

```
const colors = [ 'Violet', 'Indigo', 'Blue'
, 'Green', 'Yellow', 'Orange', 'Red',
'Blue'];

console.log(colors.indexOf( 'Blue'));
//search from the index 0 and returns 2
console.log(colors.indexOf( 'Blue',3))
//search from the index 3 and returns 7

console.log(colors.lastIndexOf( 'Blue'));
//returns 7
console.log(colors.lastIndexOf( 'Indigo'
,4));
//search backwards and returns 1
```

Slide Note

```
const colors1 = ["Violet", "Indigo",
"Blue "];
const colors2 = ["Green", "Yellow",
"Orange", "Red"];
let colors = [...colors1,
...colors2];
//Prints all the rainbow colors
console.log(colors);
//Assign the first and second item
from num array to 'one' and 'two'
variables and the rest in an array
const num = [1, 2, 3, 4, 5, 6];
const [one, two, ...rest] = num;
```

Array Spread Operator

- The JavaScript spread operator is denoted by three dots (...).
- It allows us to quickly copy all or part of an existing array or object into another array or object.
- The spread operator is often used in combination with the array destructuring.

Slide Note

Commonly Used Array Methods

Method	Description
filter()	Returns a new array containing all elements of the calling array for which the provided filtering function returns true.
indexOf()	Search the array for an element and returns its first index.
join()	Joins all elements of an array into a string.
map()	Creates a new array with the results of calling a function for each array element.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the array's new length.
reduce()	Reduce the values of an array to a single value.
reverse()	Reverses the order of the elements in an array.
shift()	Removes the first element from an array and returns that element.
slice()	Selects a part of an array and returns the new array.
sort()	Sorts the elements of an array.
splice()	Adds/Removes elements from an array.
unshift()	Adds new elements to the beginning of an array and returns the array's new length.

Slide Note

Is it possible to have functions to perform array operations so that they can be reused across different applications?

```
let salaries = [30, 40, 25, 32, 45,
28];
salaries.sort();
//[45, 40, 30, 32, 28, 25]
console.log(salaries);
```

Array Mutation

- The code shown here sorts the array containing original salaries using a built-in sort function.
- After sorting, the original list of salaries is not available for the organization to do any other computations.
- The original salary array object is mutated or changed.
- The mutation is a side effect, which can go unnoticed in large applications and can cause some hard-to-track bugs.
- Alternatively, a new array can be declared to store the incremented salaries as shown in the second code, but increases the lines of code as a result.

Slide Note

Array Immutable Methods

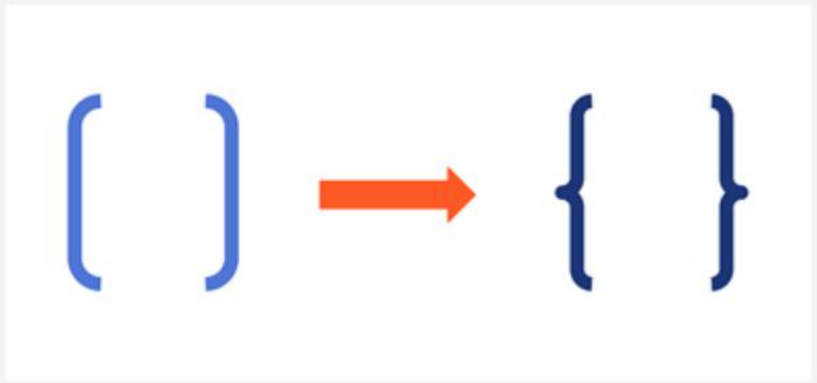
- Immutable array methods don’t mutate the original array but return a new array instead.
- The following are some of the array immutable methods.

Slide Note

Functions: First Class Citizens of JavaScript

- Functions are the first-class citizens in JavaScript.
- A programming language is said to have first-class functions if the functions in that language are treated like other variables. For example,
  - A function can be assigned as a value to a variable.
  - They can be passed as an argument for other functions.
  - They can be returned by another function.

Slide Note



Array Methods and Arrow Functions

- Array built-in methods like filter(), find(), map(), etc. take functions as arguments.
- These functions are simple and concise to create and are often better than function expressions.
- These are called arrow functions which was introduced in JavaScript ES6 version.
- We can write shorter function syntax using arrow function expressions.

Slide Note

Various Syntax in Arrow Functions

Types	General Syntax
Function with one parameter	`value => expression`
Function with multiple parameters	`(value1, value2) => expression`

Function with multi-line statements	`value => {`
	`let result = 0;`
	`return value + result;`
	`}`

Function with multiple parameters with multi-line statements	`(value1, value2) => {`
	`let result = 0;`
	`return value1 + value2 + result;`
	`}`

Arrow functions stored in variables	`let compute = (a, b) => expression;`

Slide Note

filter(), map() and reduce()

- filter(): The filter() method creates a new array with all elements that pass the test implemented by the provided function.
- map(): It creates a new array populated with the results of calling a provided function on every element in the calling array.
- reduce(): It executes a user-supplied reducer call-back function on each element of the array. It returns a single value which is the function's accumulated result.

Slide Note

1. Chaining helps in providing an output of one function as input to the other function.

2. Assists in writing compact code.

Chaining Array Functions

Function chaining is a programming strategy in JavaScript where multiple functions are called on the same object consecutively.

Slide Note

Complex Data Structures

- Working with the real-time scenarios requires handling complex data structures.
- The complexity is due to one data structure nested inside another data structure.
- Few examples of complex data structures with examples are:
  - An array of objects
  - An array as object property
  - An object as a property
- A `for...in` loop is used to iterate over object's properties.

Slide Note

Self-Check

What is the output of the code given below ?

```
const arr1 = [1, 3, 5];
const arr2 = [2, 4, 6];
arr1.concat(arr2);
console.log(arr1);
```

1. [1, 2, 3, 4, 5, 6]
2. [2, 4, 6]
3. [1, 3, 5]
4. Syntax error

Slide Note

Self-Check: Solution

What is the output of the code given below?

```
const arr1 = [1, 3, 5];
const arr2 = [2, 4, 6];
arr1.concat(arr2);
console.log(arr1);
```

1. [1, 2, 3, 4, 5, 6]
2. [2, 4, 6]
3. **[1, 3, 5]**
4. Syntax error

Slide Note

Self-Check

Given the array: `const colors = [ 'Red', 'Blue', 'Green'];`
Which operation returns the array `['Red', 'Yellow', 'Blue', 'Green']`?

1. `colors.push('yellow');`
2. `colors.unshift('yellow');`
3. `[...colors.slice(0,1),'yellow',...colors.slice(1)];`
4. `colors.splice(1,0);`

Slide Note

Self-Check: Solution

Given the array: `const colors = [ 'Red', 'Blue', 'Green'];`
Which operation returns the array `['Red', 'Yellow', 'Blue', 'Green']`?

1. `colors.push('yellow');`
2. `colors.unshift('yellow');`
3. `[...colors.slice(0,1),'yellow',...colors.slice(1)];`
4. `colors.splice(1,0);`