

LEARN THIS BEFORE REACT



```
1 // Importing  
import module  
  'modulePath'  
2  
3  
4 // Importing  
import { mod  
  'modulePath'  
5  
6  
7 // Importing  
on object
```

swipe »

7 JavaScript Concepts to Master Before React


Table of contents :

1. Arrow Functions
2. Map and Filter
3. Slice() and Splice()
4. Destructuring
5. Rest and Spread Operators
6. Template Literals
7. Promises and Async/Await



1. Arrow Functions

Arrow functions provide a concise way to write function expressions.



```
const greet = name => `Hello, ${name}!`;
console.log(greet('React Developer'));
```

Output:

```
Hello, React Developer!
```


2. Map and Filter

Map creates a new array by transforming each element, while Filter creates a new array with elements that pass a test.

```
const numbers = [1, 2, 3, 4, 5];  
const doubled = numbers.map(num => num * 2);  
const evens = numbers.filter(num => num % 2 === 0);  
console.log('Doubled:', doubled);  
console.log('Evens:', evens);
```

Doubled: [2, 4, 6, 8, 10]

Evens: [2, 4]

3. Slice() and Splice()

Slice() returns a shallow copy of a portion of an array. Splice() changes the contents of an array by removing or replacing existing elements and/or adding new elements.

```
const fruits = ['apple', 'banana', 'cherry', 'date'];  
console.log('Slice:', fruits.slice(1, 3));  
console.log('Original after slice:', fruits);  
  
const veggies = ['carrot', 'broccoli', 'spinach', 'cucumber'];  
console.log('Splice:', veggies.splice(1, 2, 'kale'));  
console.log('Original after splice:', veggies);
```

Slice: ['banana', 'cherry']

Original after slice: ['apple', 'banana', 'cherry', 'date']

Splice: ['broccoli', 'spinach']

Original after splice: ['carrot', 'kale', 'cucumber']

4. Destructuring

Destructuring allows you to extract values from objects or arrays into distinct variables.



```
const person = { name: 'John', age: 30, job: 'Developer' };  
const { name, age } = person;  
console.log(`${name} is ${age} years old.`);  
  
const colors = ['red', 'green', 'blue'];  
const [firstColor, secondColor] = colors;  
console.log(`First color: ${firstColor}, Second color: ${secondColor}`);
```

John is 30 years old.

First color: red, Second color: green

5. Rest and Spread Operators

The rest operator collects multiple elements into an array. The spread operator expands an iterable into individual elements.



```
// Rest operator
const sum = (...numbers) => numbers.reduce((acc, num) => acc + num, 0);
console.log('Sum:', sum(1, 2, 3, 4, 5));


// Spread operator
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const combined = [...arr1, ...arr2];
console.log('Combined array:', combined);
```

Sum: 15

Combined array: [1, 2, 3, 4, 5, 6]

6. Template Literals

Template literals allow embedded expressions and multi-line strings.



```
const name = 'React';  
const version = 18;  
const greeting = `Hello, ${name}!  
You are using version ${version}.  
Welcome to the world of web development.`;  
console.log(greeting);
```

```
Hello, React!  
You are using version 18.  
Welcome to the world of web development.
```


7. Promises and Async/Await

Promises and `async/await` are used for handling asynchronous operations.

```
const fetchData = () => {  
  return new Promise(resolve => {  
    setTimeout(() => resolve('Data fetched!'), 2000);  
  });  
};  
  
async function getData() {  
  console.log('Fetching data...');  
  const result = await fetchData();  
  console.log(result);  
}  
  
getData();
```

```
Fetching data...  
(after 2 seconds)  
Data fetched!
```