WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

JS →

Essential JavaScript Concepts to Master Before Diving into React

JB WEB DEVELOPER
Coding | Tutor Desishub | Youtuber

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

JS

JavaScript is the backbone of modern web development, and having a solid understanding of its core concepts is crucial before diving into React. In this **GUIDE** , we will explore the key JavaScript concepts that will lay a strong foundation for your React journey.

JB WEB DEVELOPER
Coding | Tutor Desishub | Youtuber

WATCH THE FULL VIDEO
ON YOUTUBE

Ternary Operators

Ternary operators provide a concise way to write conditional statements. They consist of three parts: a condition, a value if the condition is true, and a value if the condition is false. Consider the following example:

Powered By Desishub Coding School

```
const age = 25;
const isAdult = age >= 18 ? 'Yes' : 'No';
console.log(isAdult); // Output: Yes
```

JB WEB DEVELOPER
Coding | Tutor Desishub | Youtuber

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

Template Literals (Backticks)

Template literals, denoted by backticks (``), allow for easier string interpolation and multiline strings. They support embedding expressions within ${}`. Here's an example:

Powered By Desishub Coding School

```
const name = 'John Doe';
const greeting = `Hello, ${name}!`;
console.log(greeting);// Output: Hello, John Doe!
```

JB WEB DEVELOPER
Coding | Tutor Desishub | Youtuber

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

Destructuring (Objects and Arrays)

Destructuring simplifies the process of extracting values from arrays or objects. It enables you to unpack values into distinct variables. Here are examples of destructuring objects and arrays:

Powered By Desishub Coding School

```
// Object Destructuring
const person = { name: 'John', age: 30 };
const { name, age } = person;
console.log(name, age); // Output: John 30 //

Array Destructuring
const numbers = [1, 2, 3, 4, 5];
const [first, second, ...rest] = numbers;
console.log(first, second, rest);
// Output: 1 2 [3, 4, 5]
```

The spread operator

The spread operator in JavaScript, denoted by three consecutive dots (...), allows you to expand an iterable object, such as an array or a string, into individual elements. It provides a concise and powerful way to manipulate and combine data. The spread operator can be used in various scenarios, including array manipulation, function arguments, object literals, and more. Let's explore some examples to understand its functionality:

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

The spread operator

Combining Arrays and combining objects:

When used with an array, the spread operator allows you to create a new array by expanding the elements of an existing array. The spread operator can also be used with object literals to create new objects or merge objects together. Here's are examples:

Powered By Desishub Coding School

```
// combining Arrays
const domesticAnimals = ["Cows", "Goats", "Dogs"];
const wildAnimals = ["lions", "Kobs", "Gorillas"];
const animals = [...domesticAnimals, ...wildAnimals];
console.log(animals);

// combining Objects
const person = { name: 'John', age: 30 };
const newPerson = { ...person, city: 'New York' };
console.log(newPerson);
// Output: { name: 'John', age: 30, city: 'New York' }
```

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

Functions (Function Expression and Arrow Functions)

Functions are fundamental in JavaScript. Function expressions and arrow functions provide concise ways to define functions. Here's an example of both:

Powered By Desishub Coding School

```
// Function Expression
const square = function (num) {
    return num * num;
};
console.log(square(5)); // Output: 25

// Arrow Function
const cube = (num) => num * num * num;
console.log(cube(3)); // Output: 27
```

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

Modules (Export and Import)

JavaScript modules allow for code organization and reusability. They split code into separate files and enable exporting and importing of functionality. Consider the following example:

Powered By Desishub Coding School

```
//Module A - Exporting

export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;

//Module B - Importing

import { add, subtract } from './moduleA';

console.log(add(5, 3)); // Output: 8
console.log(subtract(10, 4)); // Output: 6
```

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

Array Methods (Map, find, filter, etc.)

JavaScript provides numerous array methods to perform common operations efficiently. Let's explore a few popular ones:

Powered By Desishub Coding School

```
const numbers = [1, 2, 3, 4, 5];

//map
const doubled = numbers.map((num) => num * 2);
console.log(doubled); // Output: [2, 4, 6, 8, 10]

//find
const found = numbers.find((num) => num === 3);
console.log(found); // Output: 3

//filter
const filtered = numbers.filter((num) => num > 3);
console.log(filtered); // Output: [4, 5]
```

JB WEB DEVELOPER
Coding | Tutor Desishub | Youtuber

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

Promises to Fetch Data

Promises simplify asynchronous operations, such as fetching data from an API. They represent the eventual completion (or failure) of an asynchronous operation. Here's an example:

Powered By Desishub Coding School

```
fetch('https://api.example.com/data')
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.error(error));
```

JB WEB DEVELOPER
Coding | Tutor Desishub | Youtuber

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

Async/Await to Fetch Data

Async/await is a more modern and readable approach for handling asynchronous operations. It allows writing asynchronous code in a synchronous style. Consider the following example:

Powered By Desishub Coding School

```
async function fetchData() {
 try {
  const response = await fetch('https://api.example.com/data');
  const data = await response.json();
  console.log(data);
 } catch (error) {
    console.error(error);
   }
}
 fetchData();
```

WATCH THE FULL VIDEO
ON YOUTUBE

SUBSCRIBE

Installing Packages Using NPM

NPM (Node Package Manager) is the default package manager for JavaScript. It provides a vast collection of libraries and tools to extend your applications.

NPM (Node Package Manager) comes bundled with Node.js, so when you install Node.js, NPM is automatically installed along with it. **A package.json** file is used to manage your project's dependencies, scripts, metadata, and more

Powered By Desishub Coding School

```
// Check if npm is Installed
npm -v

//Initializing a package.json
npm init

//Install a package
npm install package-name

//example
npm install package-name
```

JB WEB DEVELOPER
@desishub

WATCH THE FULL VIDEO

JB WEB DEVELOPER

JB WEB DEVELOPER
Coding | Tutor Desishub | Youtuber