Table of Contents

[Components 1](#_Toc160898711)

[Course Introduction 1](#_Toc160898712)

[i. Introduction to Course 1](#_Toc160898713)

[Rendering Lists in React 1](#_Toc160898714)

[Forms in React 1](#_Toc160898715)

[React Context 1](#_Toc160898716)

[React Hooks & Custom Hooks 1](#_Toc160898717)

[Getting Started With Hooks 1](#_Toc160898718)

[Rules of Hooks & Fetching Data with Hooks 1](#_Toc160898719)

[Advanced Hooks 1](#_Toc160898720)

# Components

## Course Introduction

### Introduction to Course

* Course focuses on advanced React concepts and techniques.
* Intended for learners familiar with React basics.

### React and Career Opportunities

#### React's Significance in Tech Landscape:

* React has emerged as a pivotal framework in software development.
* Many job opportunities now require proficiency in React.

#### Professional Experience with React:

* React is an integral part of work at Meta.
* It powers various elements across Meta's products like Facebook and Instagram.

#### Initial Impression and Benefits of React:

* React's capabilities simplify web application development.
* It reduces errors and streamlines tasks for engineers.

#### React's Rise in Popularity:

* React has surpassed older frameworks like jQuery in search popularity.
* Its ability to keep pace with industry demands makes it highly sought-after.

#### Core Language and Variants:

* React primarily utilizes JavaScript, with variants like Flow and TypeScript.
* Different dialects offer options for type safety and development preferences.
* Typescript flavour of React is most popular in industry.

#### Versatility Across Platforms:

* React extends beyond web applications to mobile and virtual reality (VR) development.
* Its consistency across platforms facilitates innovation and development.

#### Essential Skill in Job Market:

* React proficiency is now a prerequisite for many job roles.
* Mastery of React enhances career prospects and enables scalable web application development.

#### Advice for Aspiring Developers:

* Encouragement to persevere (persist) through initial challenges in learning React.
* Consistency and dedication in learning will yield long-term benefits in career growth and project success.

## Rendering Lists in React

### Transforming lists in JavaScript

The most common type of data that we will encounter in an array/list is objects.

#### The Map() method:

In JavaScript, the map() method is used to iterate over an array and execute a provided function on each element of the array. It then returns a new array containing the results of applying the function to each element. The original array remains unchanged.

The syntax for map() is as follows:

array.map(callback(currentValue, index, array), thisArg)

#### filter() method

In JavaScript, the filter() method is used to create a new array with elements that pass a specified test implemented by a provided function. It iterates over each element in the array and returns a new array containing only the elements for which the provided function returns true.

const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

const oddNumbers = numbers.filter(function(number) {

return number % 2 !== 0;

});

The main difference between filter() and map() is that filter() is used to create a new array with elements that pass a specified condition, while map() is used to create a new array by applying a function to each element in the original array.

#### reduce() method:

The reduce() method in JavaScript is used to reduce the elements of an array to a single value. It executes a provided function on each element of the array and accumulates the result into a single value. The syntax for reduce() is as follows:

array.reduce(callback(accumulator, currentValue, index, array), initialValue)

callback: A function that is called once for each element in the array. It takes four arguments:

accumulator: The accumulator accumulates the callback's return values. It is the accumulated value previously returned in the last invocation of the callback or the initialValue, if supplied.

currentValue: The current element being processed in the array.

index (optional): The index of the current element being processed.

array (optional): The array reduce() was called upon.

**Example:**

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

const sum = numbers.reduce(function(accumulator, currentValue) {

return accumulator + currentValue;

}, 0);

#### sort() method

The sort() method in JavaScript is used to sort the elements of an array in place and returns the sorted array. By default, the sort() method sorts elements alphabetically (for strings) and numerically (for numbers). The sorting is done based on the Unicode values of the elements.

The syntax for sort() is simple:

array.sort()

The sort() method sorts the elements of the array in place and returns the sorted array. The original array is modified, and no new array is created.

However, you can also provide a custom sorting function as an argument to sort(), which defines the sorting order based on your requirements.

array.sort(compareFunction)

const words = ["apple", "banana", "orange", "grapefruit"];

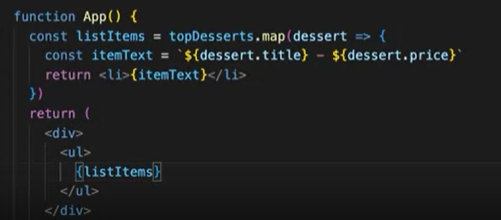
words.sort(function(a, b) {

return a.length - b.length;

});

### Render a simple list component

All HTML elements are ‘components’ by default.



### What are keys in React

## Forms in React

## React Context

# React Hooks & Custom Hooks

## Getting Started With Hooks

## Rules of Hooks & Fetching Data with Hooks

## Advanced Hooks