Difference between Props & state?

for Parent-child communication, simply pass props

Use state to store the data your current page needs in your controller-view.

Use props to to pass data & event handlers down to your child components.

These lists should help guide you make working the data in your components.

Props

- are mutable
  - which lets React do fast reference che
- are used to pass data down from view-controller
  - Your top level component

- have better performance
  - use this to pass data to ~~world~~ child components

state
- should be managed in your view-controller
  - your top level component
- is mutable
- has worse performance
- should be accessed from child components.
  - pass it down with props instead.

- What is useState() API

The useState() is a Hook that allows you to have state variables in functional components. So basically useState is the ability to encapsulate local state in a functional component.

- How map, filter, reduce work?

→ How to use map() Method.

Suppose you have an array arrOne where you've stored some numbers, and you'd like to perform some calculations on each of them. But you also don't want to mess with the original array.

This is where map() comes into picture. The map method will help you do this

```
let arrOne = [32, 45, 63, 36, 24, 11];
```

map() takes the maximum of three arguments which are value / element, index, and array.

```
arrOne.map(value/element, index, array)
```

Lets say you want to multiply each element by 5 while not changing the original array.

Here's the code to do that:

```
let arrOne = [32, 45, 63, 36, 24, 11]
const multFive = (num) => {
    return num * 5;
}
let arrTwo = arrOne.map(multiFive)
console.log(arrTwo)
```

And here's the output:

```
[160, 225, 325, 180, 120, 55];
```

So what's going on here? well, I have an arrOne array with 6 elements in it. Next, I initialized an arrow function multFive with 'num' as an argument. This returns the product of num and 5, where the 'num' variable is fed the data by the map() method.

~~function (num)~~
~~{~~
~~return num * 5;~~
~~}~~

→ How to use filter() method.

The name kind of gives it away, doesn't it? You use this method to filter the array based on conditions you provide. The filter() method also creates a new array.

Let's take a Example: Suppose you have an array arrNum and that array stores a bunch of numbers. Now, you would like to see what numbers can be divided by 3 and make a separate array from them.

Here's the code to do that:

```
let arrNum = [15, 39, 20, 32, 30, 45, 22]

function divByFive(num) {
    return num % 3 == 0
}

let arrNewNum = arr.Num.filter(divByFive)

Console.log(arrNewNum)
```

And here's the output:

```
[15, 39, 30, 45];
```

Let's break down this code. Here I have an array arrNum with 7 elements in it. Next, I initialized a function divBy five with 'num' as an argument. It returns true or false for each time a new num is passed for the comparision, where the 'num' variable is fed the data by the filter() method.

→ How to use Reduce() Method:

Let's say you are asked to find the sum of all elements of an array. Now, you could use a for loop or the forEach() method, but reduce is built for this kind of task.

The reduce() method reduces an array to a single value by performing the desired operation on the elements collectively.

Let's take the above example and use reduce on it:

```
let arrNum = [15, 39, 20, 32, 30, 45, 22]
    function sumofEle(num, ind) {
        return num+ind;
    }
let arrNum2 = arrNum.reduce(sumOfEle)
console.log(arrNum2);
```

Here's the output

DOS

Everything is the same as the map() and filter() methods - but what's important to understand is how the reduce method works under the hood.

There's not a definite syntax of the reduce method. Let's see the simplest one and it will give you the gist of all the ways you can use reduce().

Here's some example syntax for reduce

let arrNum = [15, 39, 20, 32, 30, 45, 22] arr.reduce

((a1, a2) => { return a1 + a2

});