* In Browser two engines
  ① Execution Engine (Javascript)
  ② Rendering Engine (HTML & CSS)

* React: to not rerender other components

* npm : node package manager
  npx : node package executor

```
function setState(){
    let score = 22
    return score
}

let myScore = setState()
myScore;
myScore = 12
console.log(myScore);

let anotherScore = setState()
anotherScore;

console.log(anotherScore);
```

* here we are changing the copy of score and not the actual score as function is passing a copy of store

* Primitive Datatypes
  Number, string, Boolean
  (we get a copy of them and the actual value does not change)

* Non Primitive Datatypes
  Array & Object
  (We call them by reference and value gets changed)

* in App.js

```
function counter() {
  return (
   <div>
      <button> Count : 1 </button>
   </div>
  )
}
function App() {
   return (
    <>
   <h1> Hello to react </h1>
   < Counter /> < Counter /> <Counter />
    </>
   );
 }
export default App;
```

Virtual DOM is creating copies in the same memory location

```
* import {useState} from 'react'
  function counter() {
     let [count, setCount] = useState(100);
                 variable        value of count
   function onClip() {
     setCount(count+1) ;
   }
    return (
    <div>
       <button onclick="onClip" > Count : {count} </button>
   </div>
  )
}
```

```jsx
import {useState} from 'react'

function SuperHeros(){
  const [hero, setHero] = useState(["Superman", "Spiderman",
  "Ironman"])
  const [name, setName] = useState ( ()=> "antman");
  return(
    <div>
      <ul>
        {hero.map((h) => (
          <li>{h}</li>
        ))}
      </ul>
    </div>
  )
```

we used normal brackets so that we dont have to write return if { } than we have to write return

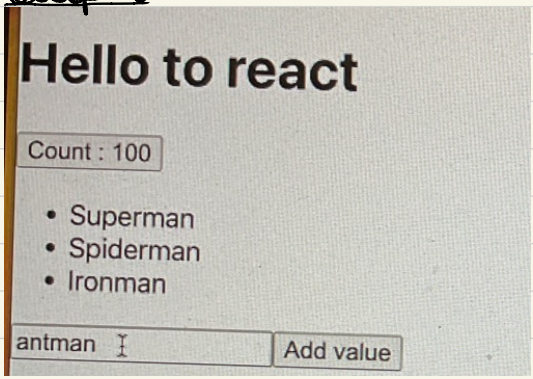
<li key={h}>{h}

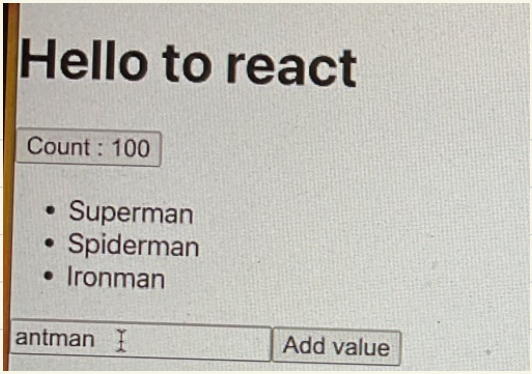
we use key so that react knows we are not looping through the same element only

```jsx
    <div>
      <ul>
        {hero.map((h) => (
          <li key={h}>{h}</li>
        ))}
      </ul>
      <input
      type="text"
      value={name}

      />
      <button>Add value</button>
    </div>
  )
}
```

Output






```
import React,{useState}
from "react";

function App() {

const [Hero, SetHero] =
  useState([]);
const [addHero, SetAddhero] = useState("");


function handleChange(event){
  return(setAddHero(event.target.value);)
}
 function Submit(){
  setHero((list)=>{[...list, addHero]};
  setAddHero("");
}
return(
  <>
  <h1> Hello to React </h1>
  {Hero.map((item)=>{
      return(<li>{item}</li>)
   });
  }
<input value={Hero} onChange={handleChange} />
<button onClick={submit}>Add Value</button>
}
```

```jsx
import {useState} from 'react'

function SuperHeros(){
  const [hero, setHero] = useState(["Superman", "Spiderman",
  "Ironman"])
  const [name, setName] = useState(() => "antman")

  const onAddName = () => {
    setHero([...hero, name])
    setName("")
  }
  return(
    <div>
      <ul>
        {hero.map((h) => (
      <li key={h}>{h}</li>
    ))}
  </ul>
  <input
  type="text"
  value={name}
    onChange={(e) => setName(e.target.value)}
  />
  <button
  onClick={onAddName}
        >Add value</button>
    </div>
  )
}
```

→ after submitting input area blank

grab the event