**React Training**

**Assignment 1- Explore Next Generation JavaScript Functions.**

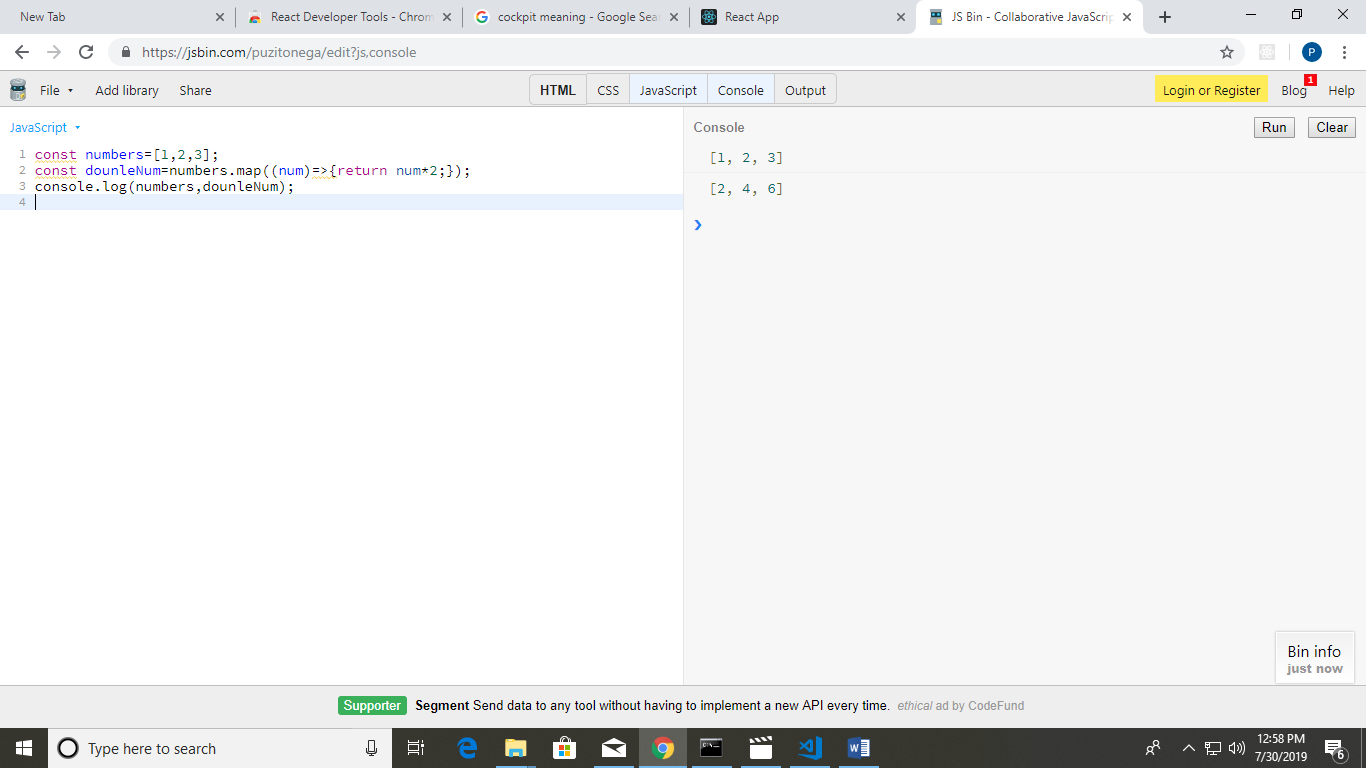
---------------------------------------------------------------------------------------------------------------------------------

**Map() function:**

The map() method creates a new array with the results of calling a function for every array element.

The map() method calls the provided function once for each element in an array, in order.

Js file Code:

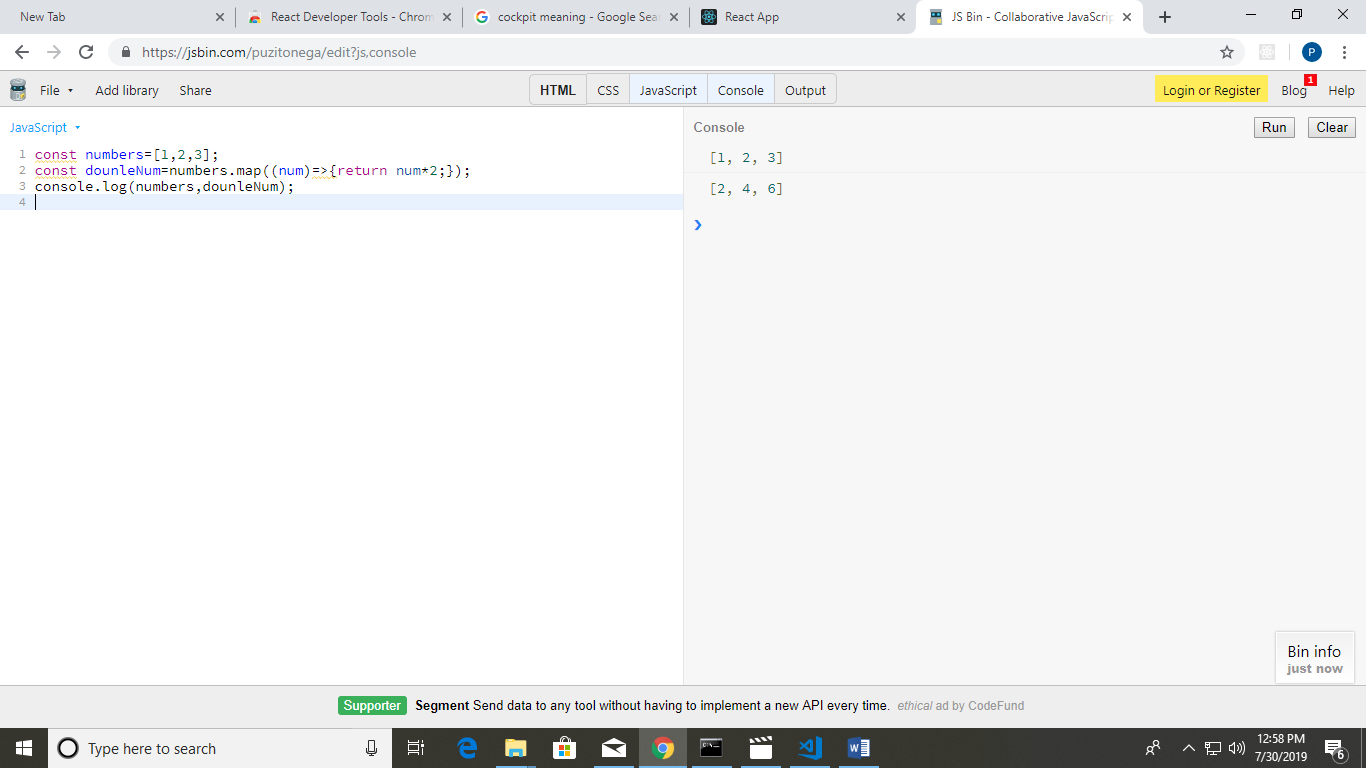


const numbers=[1,2,3];

const dounleNum=numbers.map((num)=>{return num\*2;});

console.log(numbers,dounleNum);

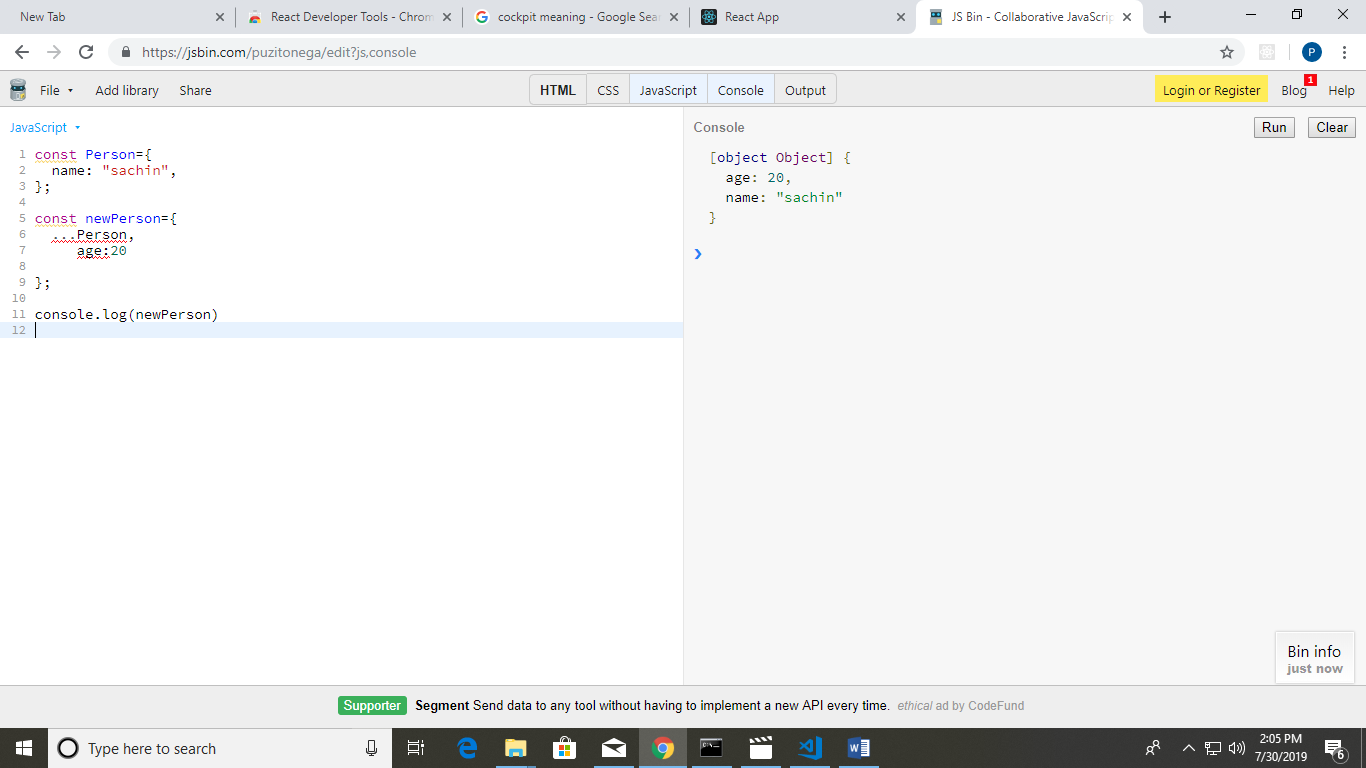
**Output:**



**Spread Operator:**

**Spread operator** allows an iterable to expand in places where 0+ arguments are expected. It is mostly used in variable array where there is more than 1 values are expected.It allows us the privilege to obtain a list of parameters from an array. Syntax of Spread operator is same as [Rest parameter](https://www.geeksforgeeks.org/javascript-rest-operator/) but it works completely opposite of it.

Js code:



const Person={

name: "sachin",

};

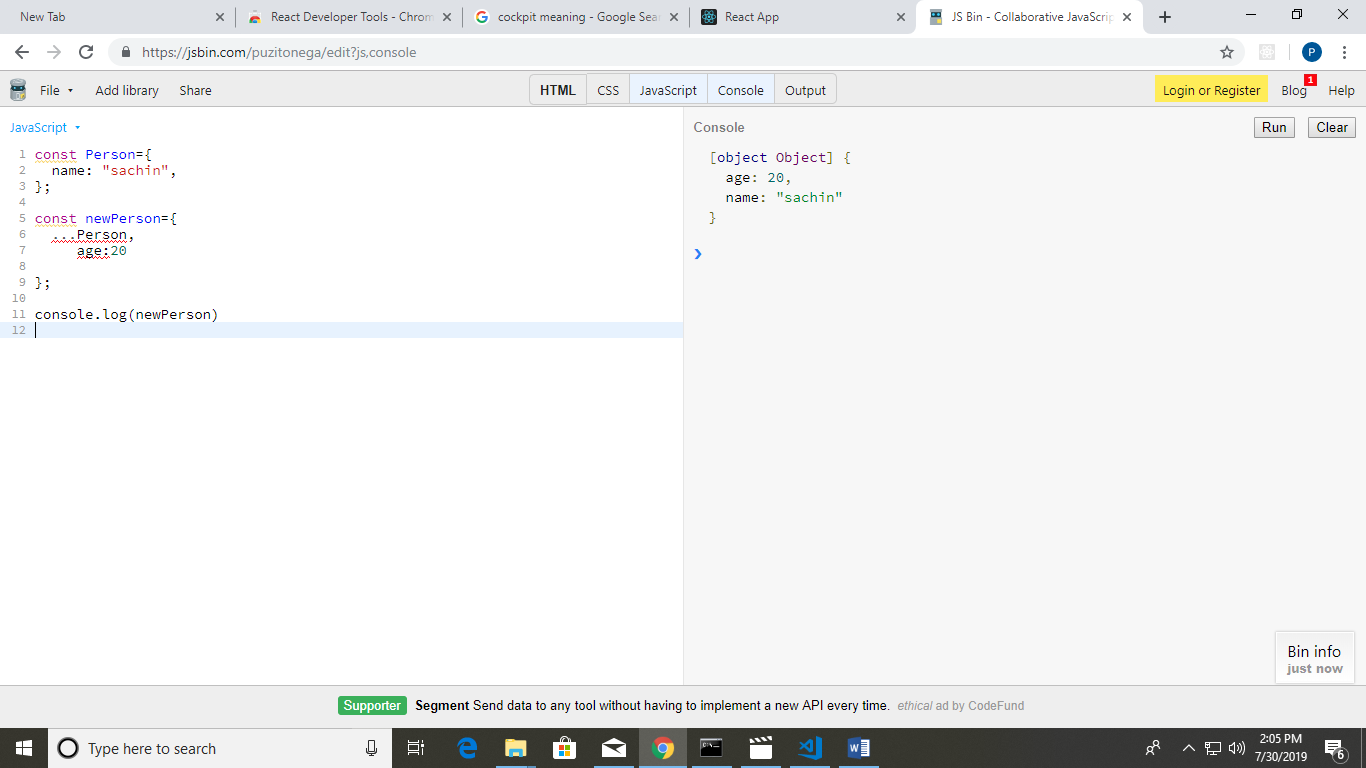
const newPerson={

...Person, age:20

};

console.log(newPerson)

Output:

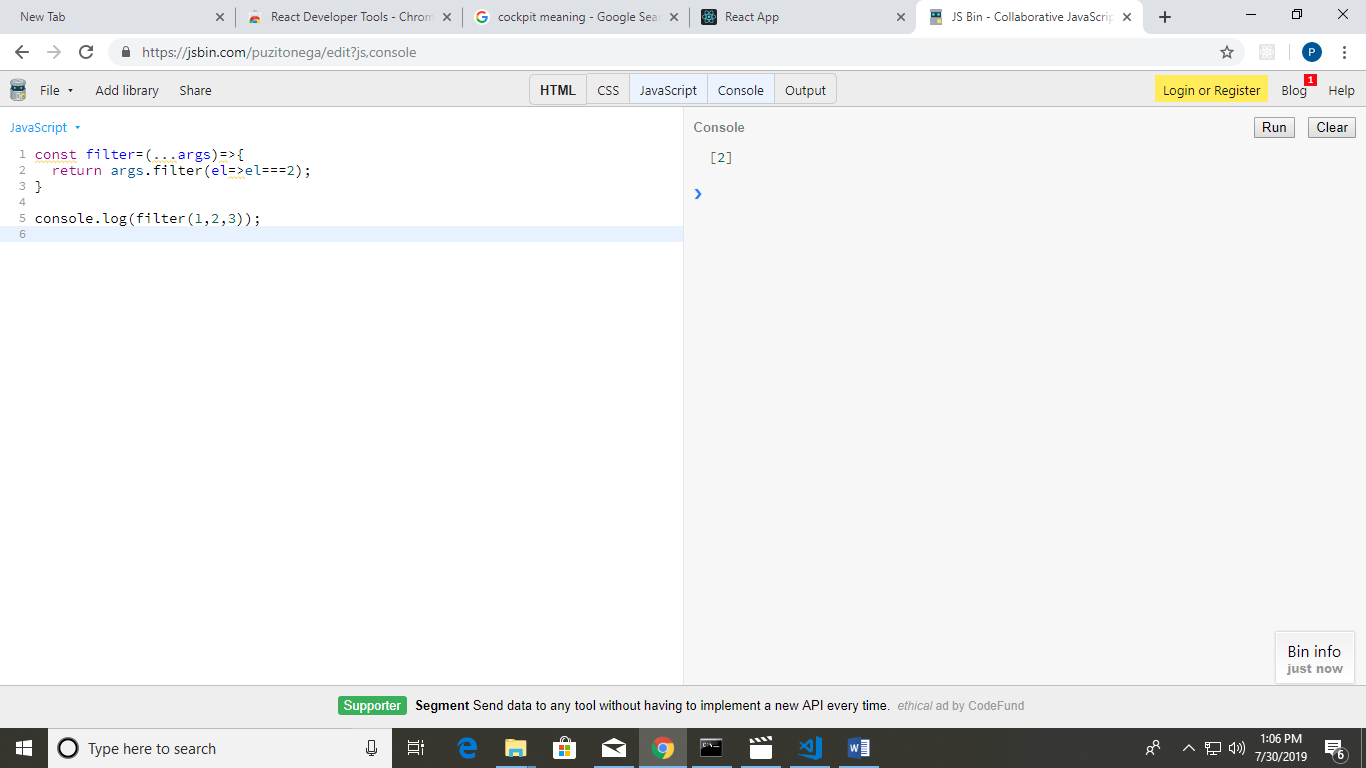


-------------------------------------------------------------------------------------------------------------

**Rest Operator:**

**Rest parameter** is an improved way to handle function parameter, allowing us to more easily handle various input as parameters in a function. The rest parameter syntax allows us to represent an indefinite number of arguments as an array. With the help of a rest parameter a function can be called with any number of arguments, no matter how it was defined. Rest parameter is added in ES2015 or ES6 which improved the ability to handle parameter.

Js file code :



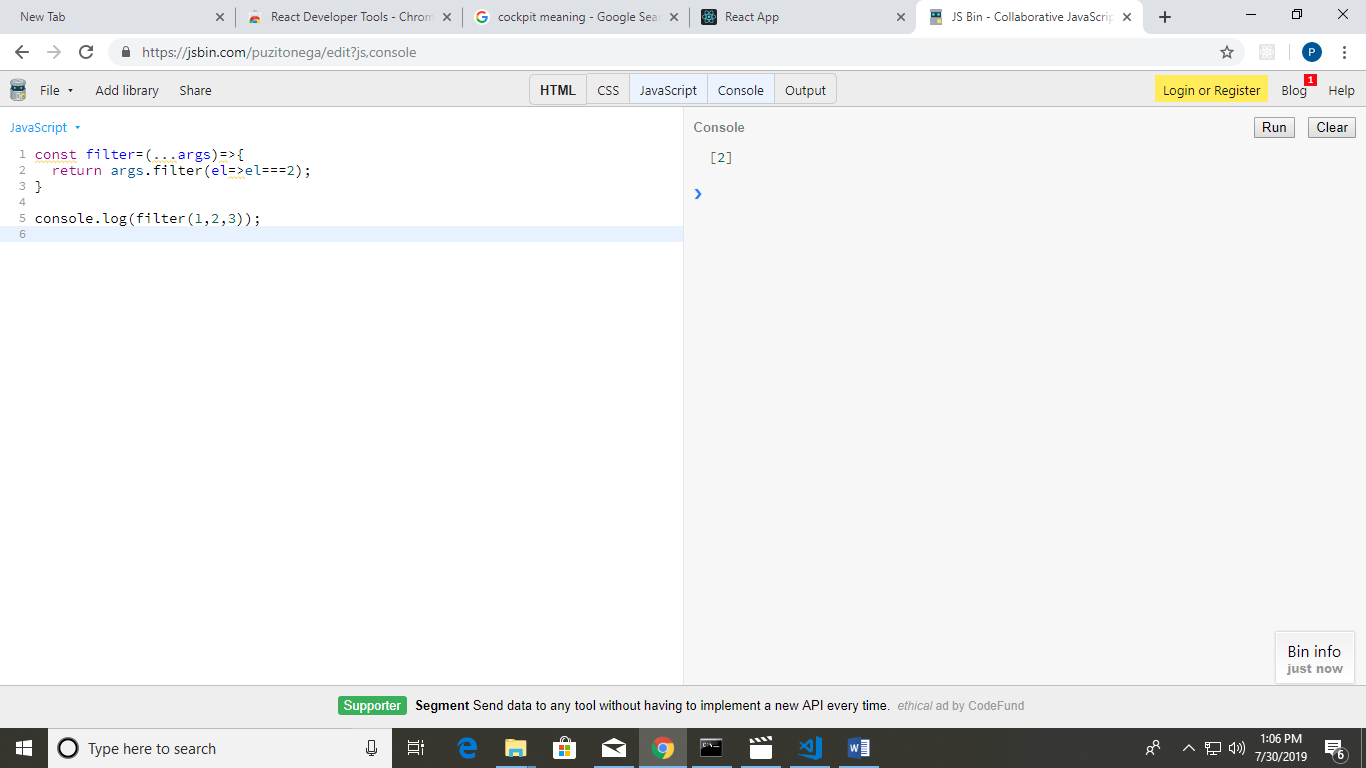
const filter=(...args)=>{

return args.filter(el=>el===2);

}

console.log(filter(1,2,3));

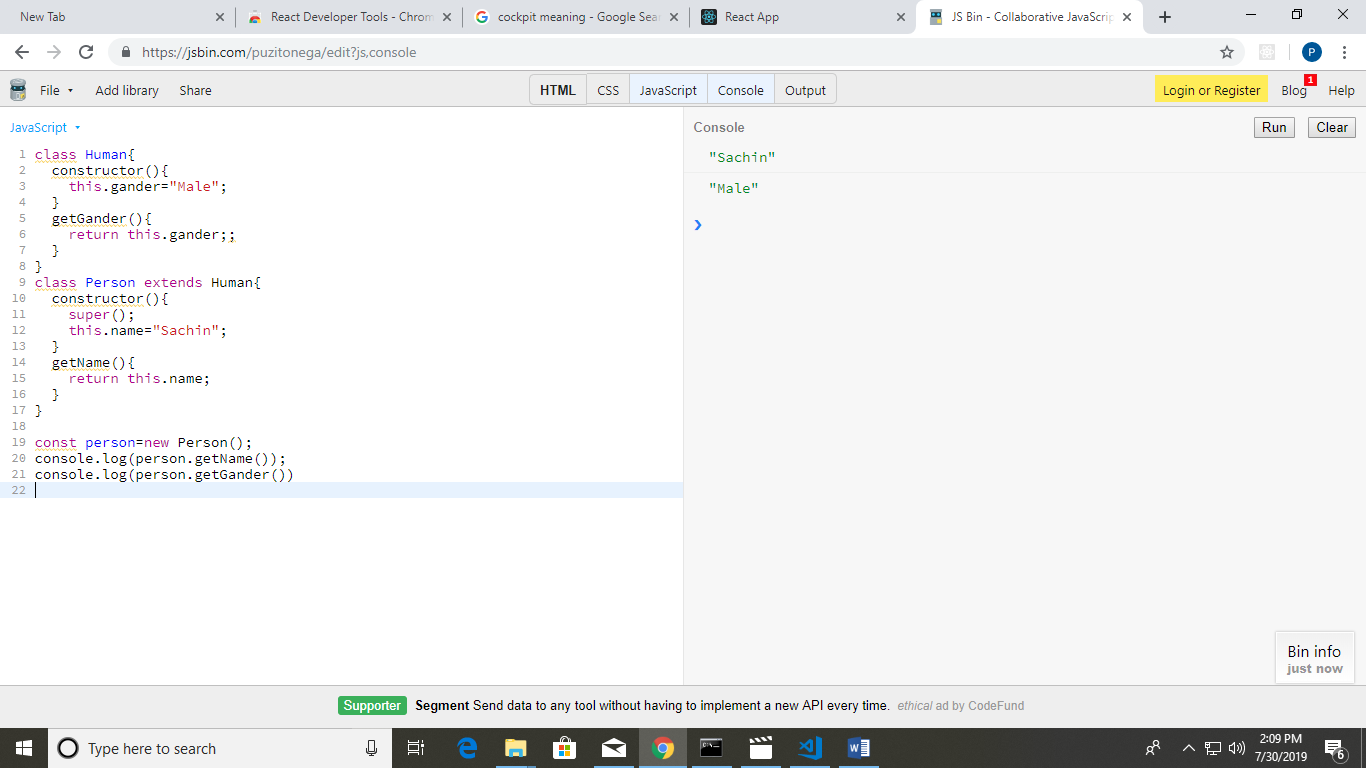
**Output:**



-------------------------------------------------------------------------------

**Class concept:**

Code for demonstration on class in java script:



class Human{

constructor(){

this.gander="Male";

}

getGander(){

return this.gander;;

}

}

class Person extends Human{

constructor(){

super();

this.name="Sachin";

}

getName(){

return this.name;

}

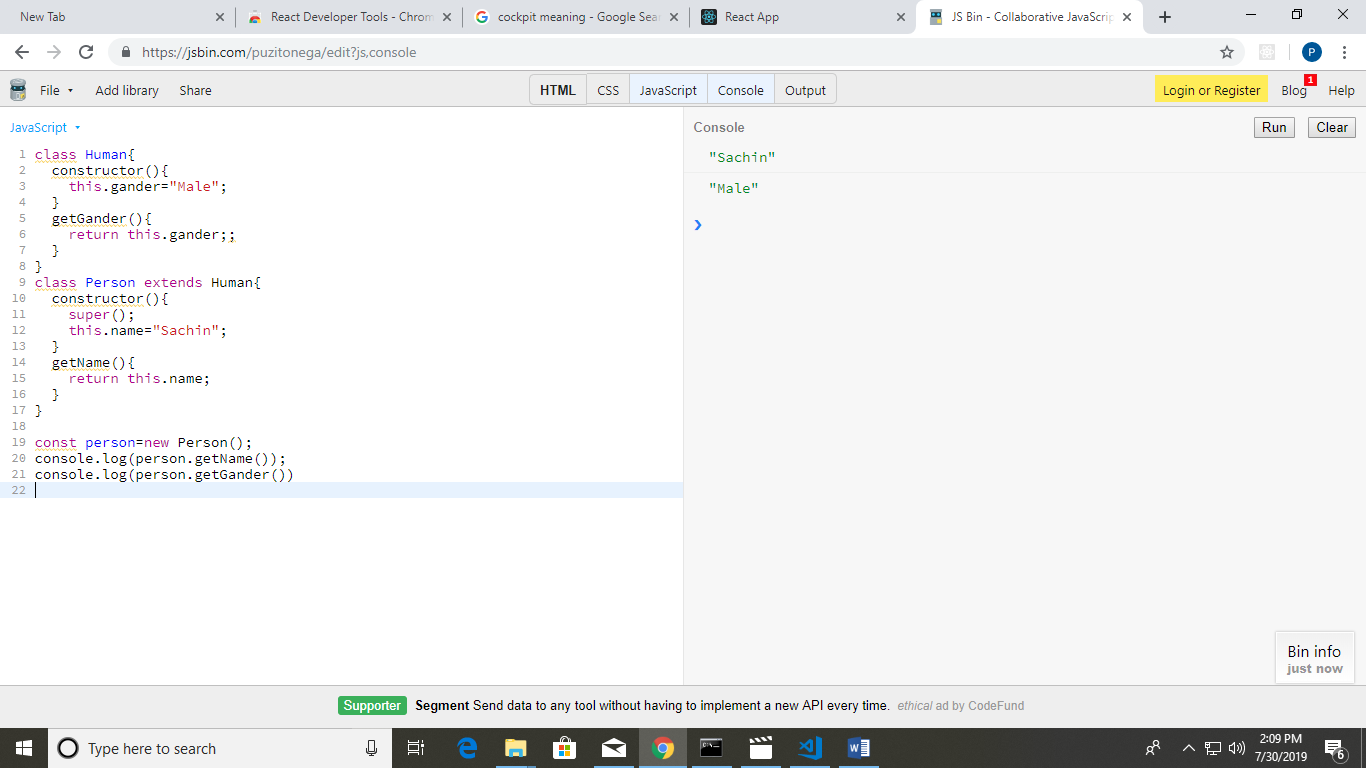
}

const person=new Person();

console.log(person.getName());

console.log(person.getGander())

**output:**

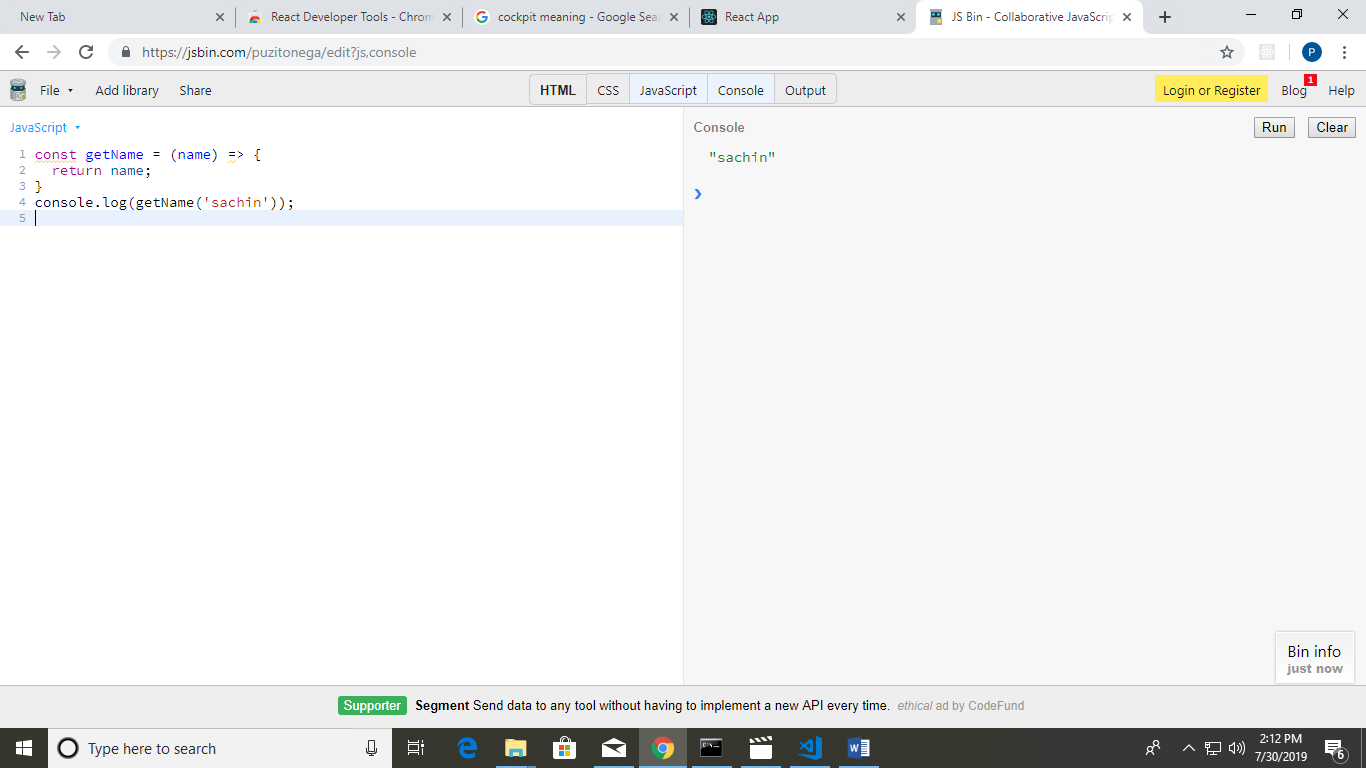


--------------------------------------------------------------------------------------------------------------------------

**Arrow Function:**

An **arrow function expression** is a syntactically compact alternative to a regular [function expression](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function), although without its own bindings to the [this](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this), [arguments](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments), [super](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/super), or [new.target](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new.target) keywords. Arrow function expressions are ill suited as methods, and they cannot be used as constructors.

**Js code:**



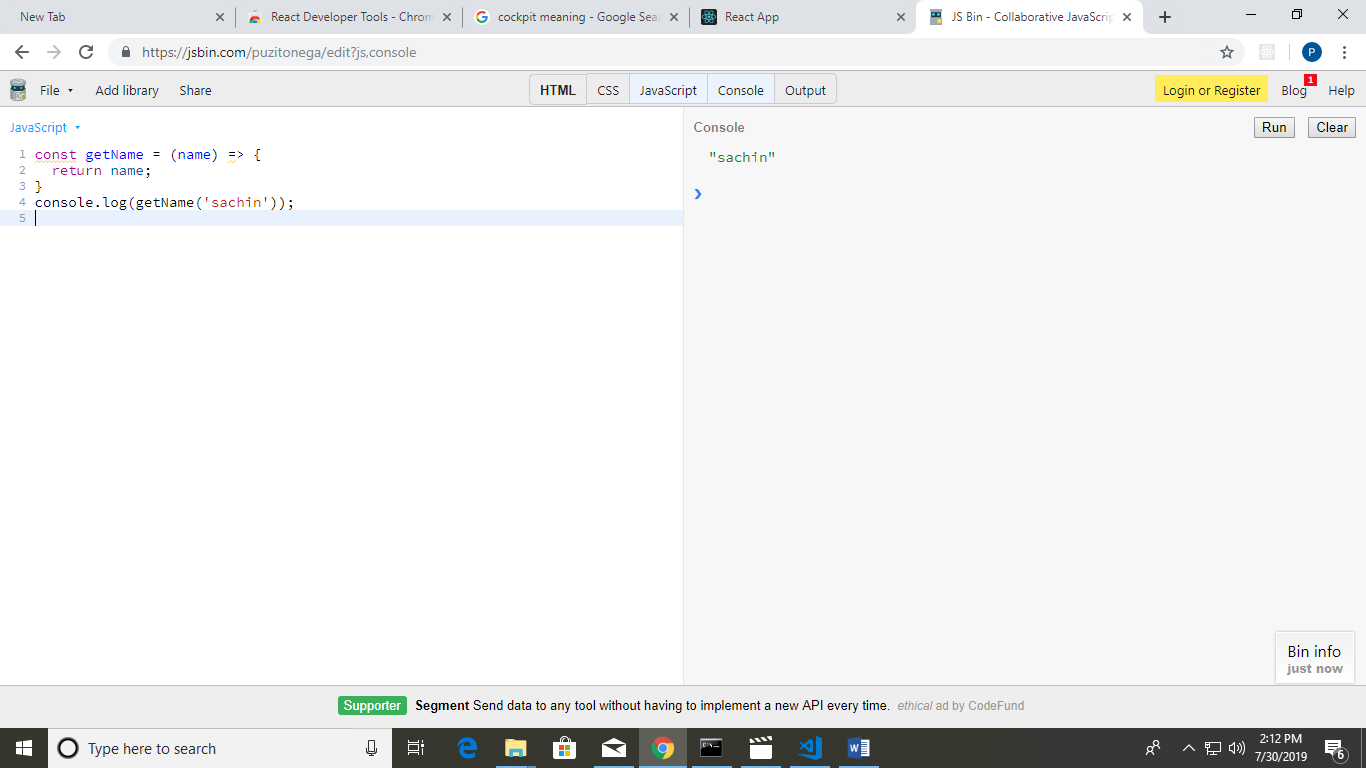
const getName = (name) => {

return name;

}

console.log(getName('sachin'));

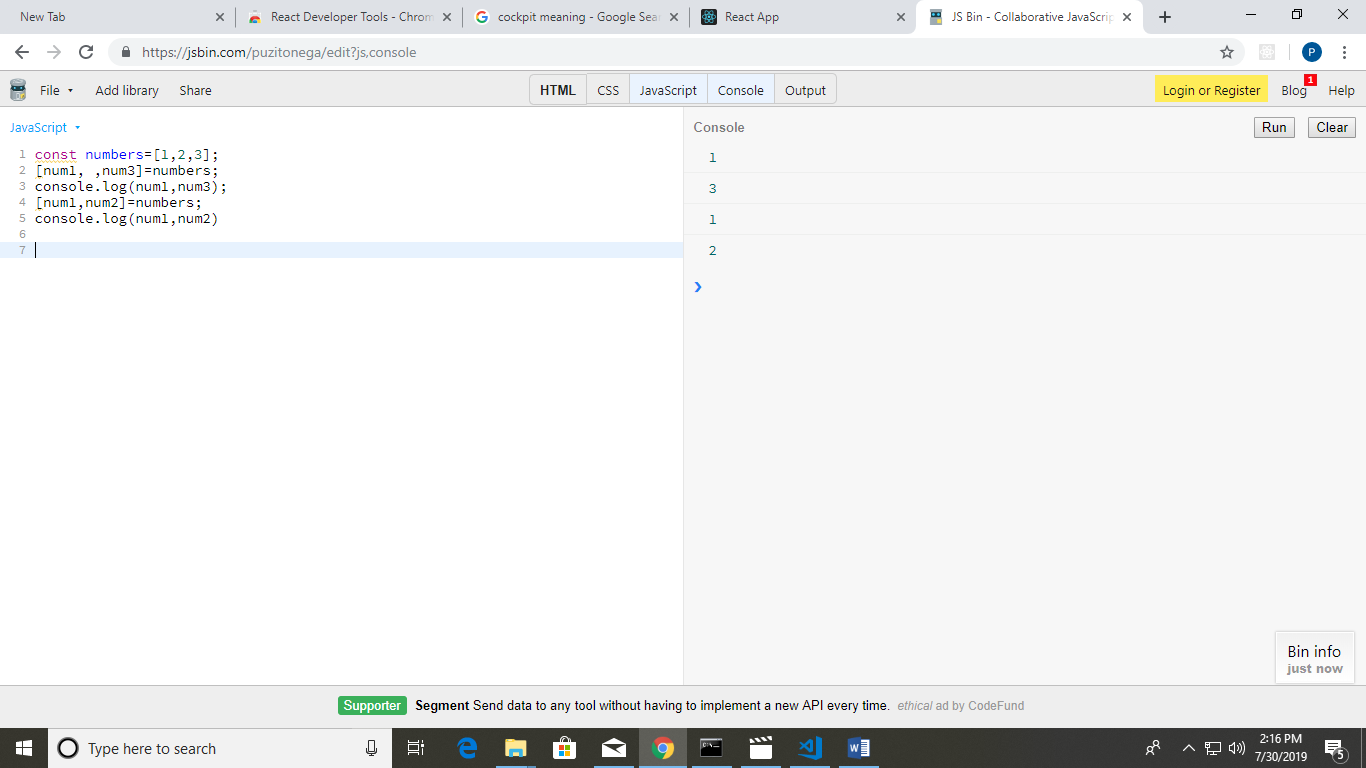
**Output:**



**Destructuring:**

Destructuring assignment is a special syntax that allows us to “unpack” arrays or objects into a bunch of variables, as sometimes that’s more convenient. Destructuring also works great with complex functions that have a lot of parameters, default values, and so on.

Code:



const numbers=[1,2,3];

[num1, ,num3]=numbers;

console.log(num1,num3);

[num1,num2]=numbers;

console.log(num1,num2)

**Output:**

