REACTJS – MODULE2

INTRODUCTION TO ES6

accenture

AGENDA

- Introduction to ES6
- Variables
- Arrow function
- Object Destructuring
- Classes
- inheritance

Introduction to ES6 (ECMA Script6)

ES6 is also known as ES2015

Is the latest javascript specifications.

Developed by a company Ecma

Released in June 2015

This is used to create ReactJS applications

accenture

New Features of ES6

- Lexical scoping using let and const keywords
- Classes and Inheritance
- Arrow function
- Default value for function
- Destructuring
- Spread operator

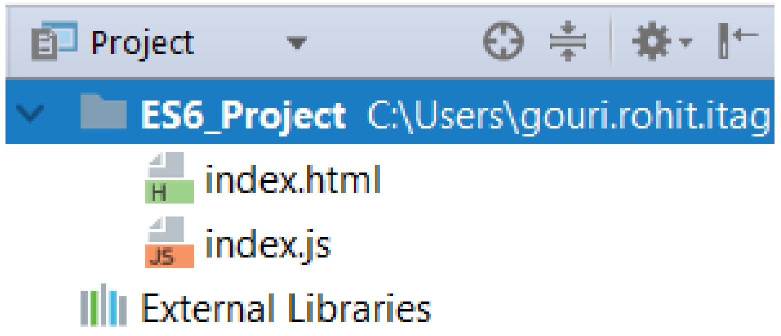
. . . And many more

Activity - Setup Project

Step 1 → Open Webstorm IDE and create new project “ES6_Project”

Step 2 → Right click on the project add new javascript file “index.js”

Step 3 → Right click on the project add new HTML file “index.html”



Project

ES6_Project C:\Users\gouri.rohit.itag

index.html

index.js

External Libraries

accenture

Activity - Setup Project

Step 4 → Open index.js file and write below code

```
document.write("Hello world from javascript file");
```

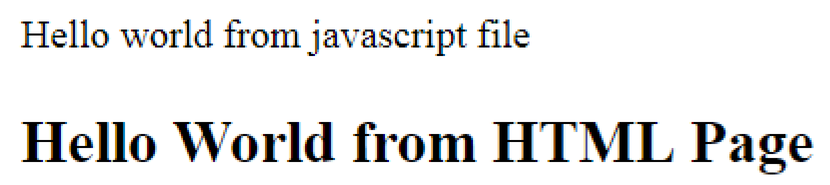
Step 5 → Open index.html file and write below code which also link index.js file

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="index.js"></script>
</head>
<body>
<h2>Hello World from HTML Page</h2>
</body>
</html>
```

accenture

Activity - Setup Project

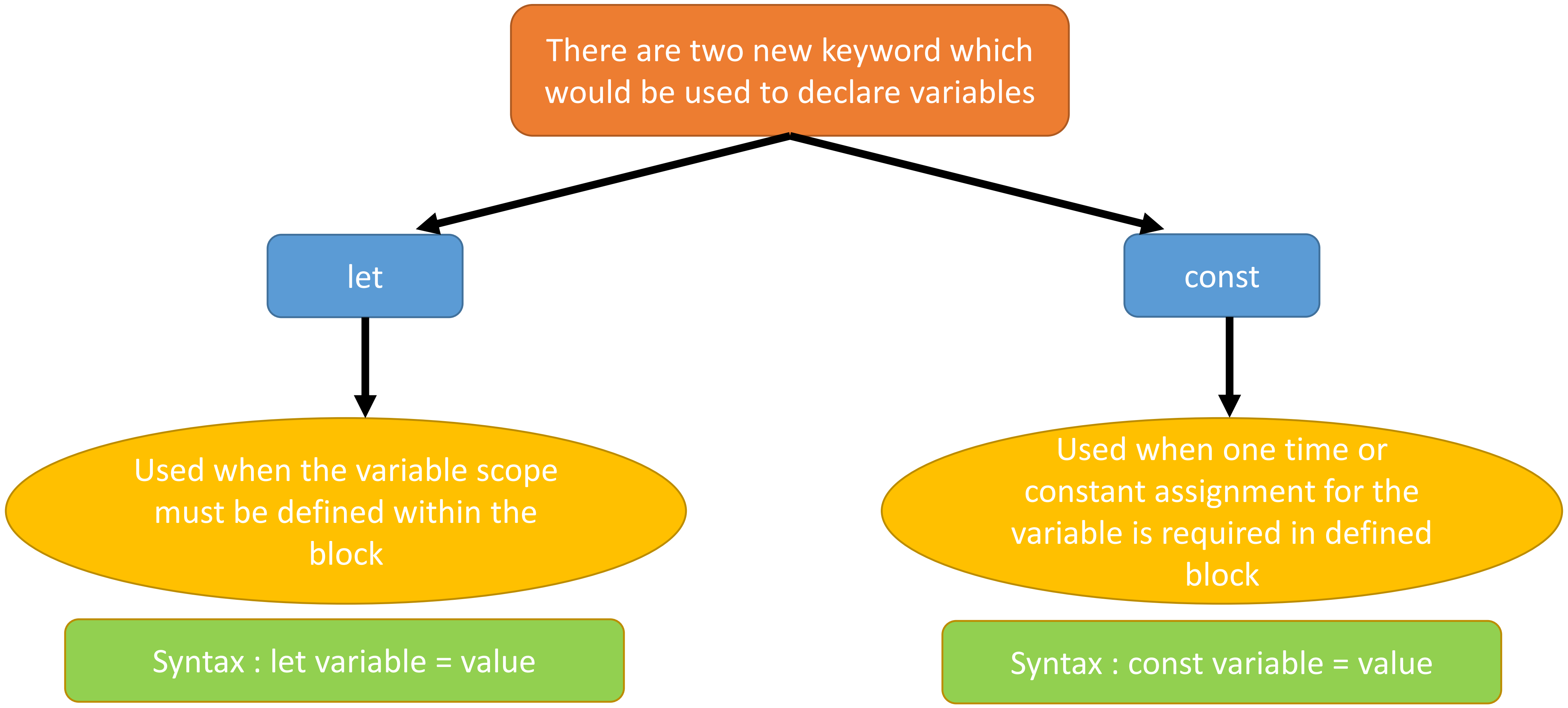
Step 6 → Run index.html in browser to see the output



```
Hello world from javascript file
```

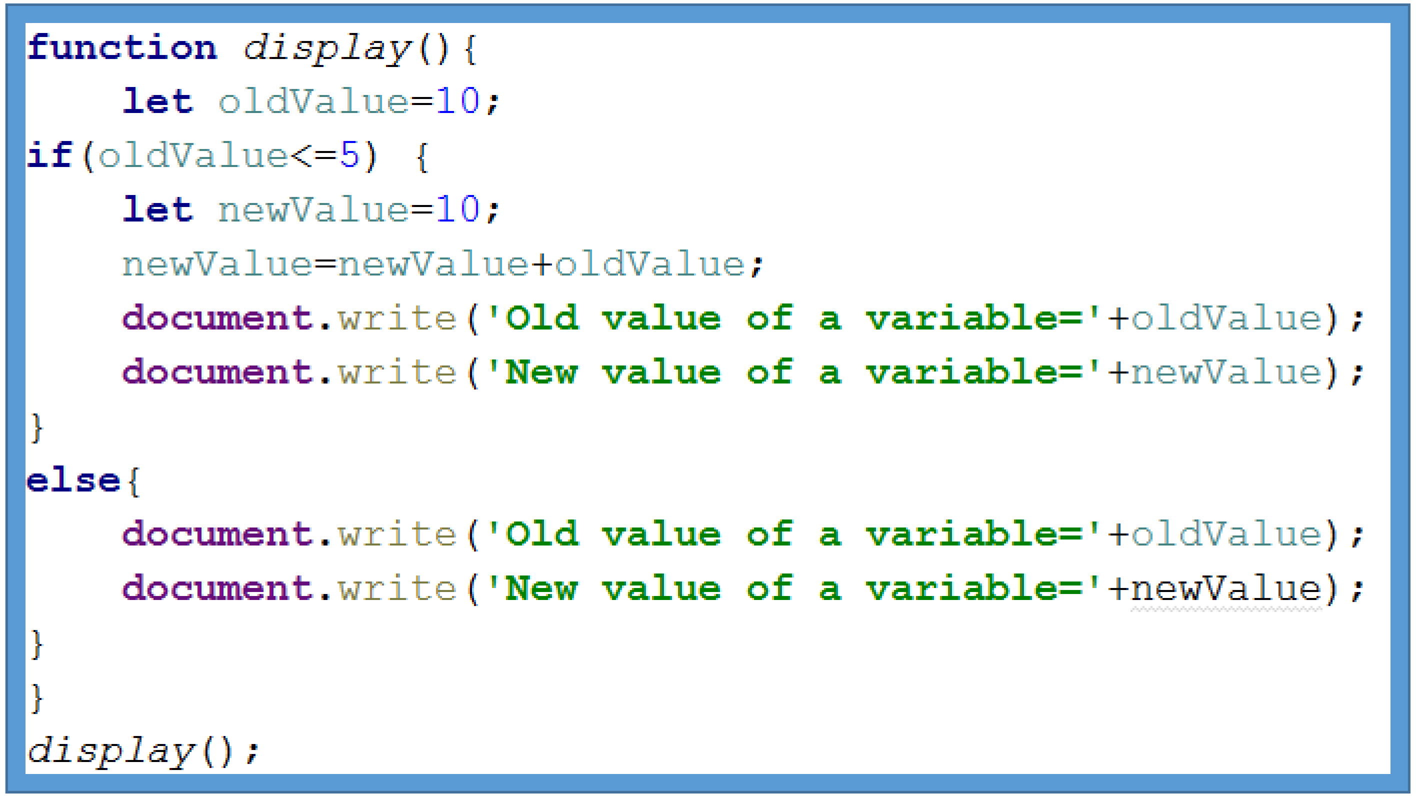
Hello World from HTML Page

ES6 New Feature - Variable


There are two new keyword which would be used to declare variables
let
const
Used when the variable scope must be defined within the block
Used when one time or constant assignment for the variable is required in defined block
Syntax : let variable = value
Syntax : const variable = value

Demo - let

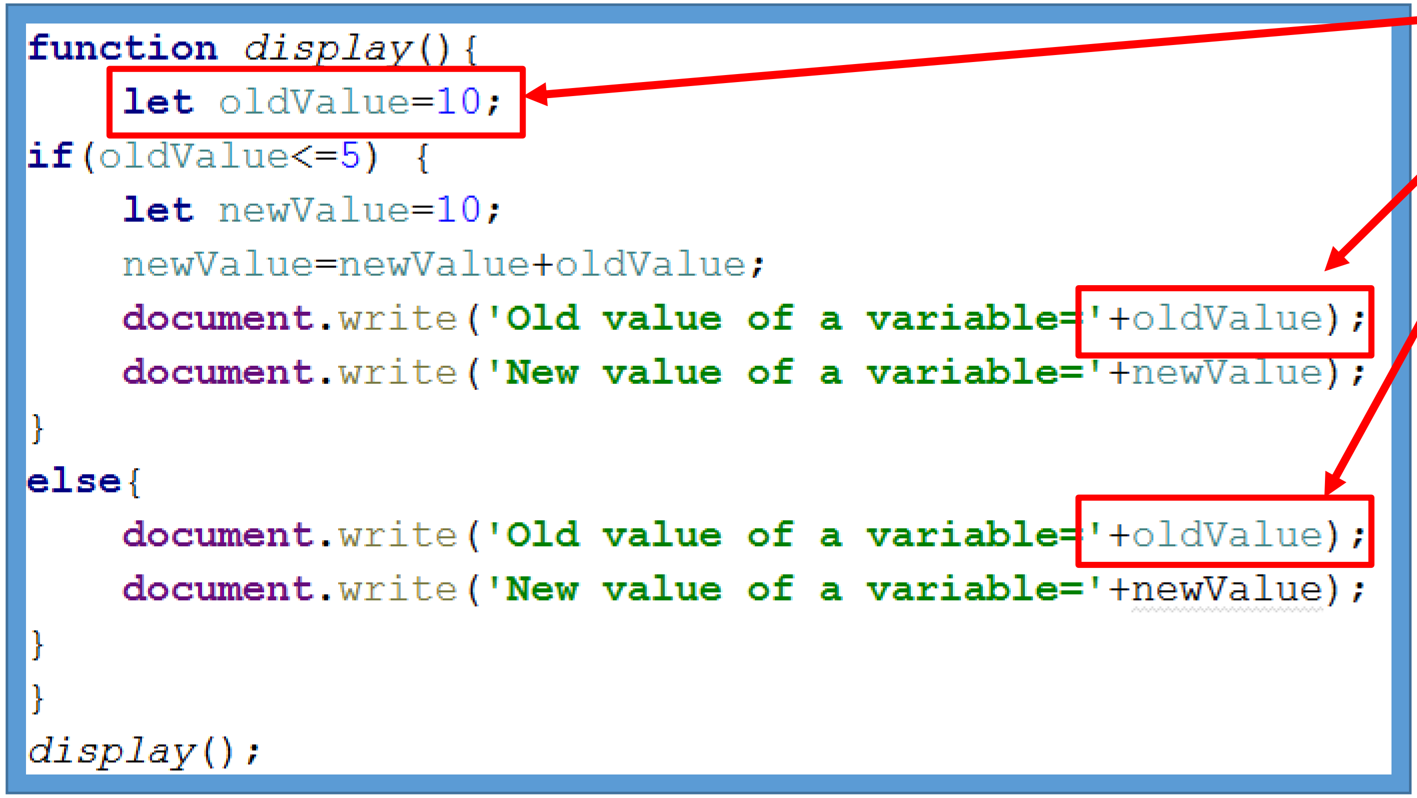
index.js



```
function display(){
    let oldValue=10;
if(oldValue<=5)  {
    let newValue=10;
    newValue=newValue+oldValue;
    document.write('Old value of a variable='+oldValue);
    document.write('New value of a variable='+newValue);
}
else{
    document.write('Old value of a variable='+oldValue);
    document.write('New value of a variable='+newValue);
}
}
display();
```

Demo - let

index.js



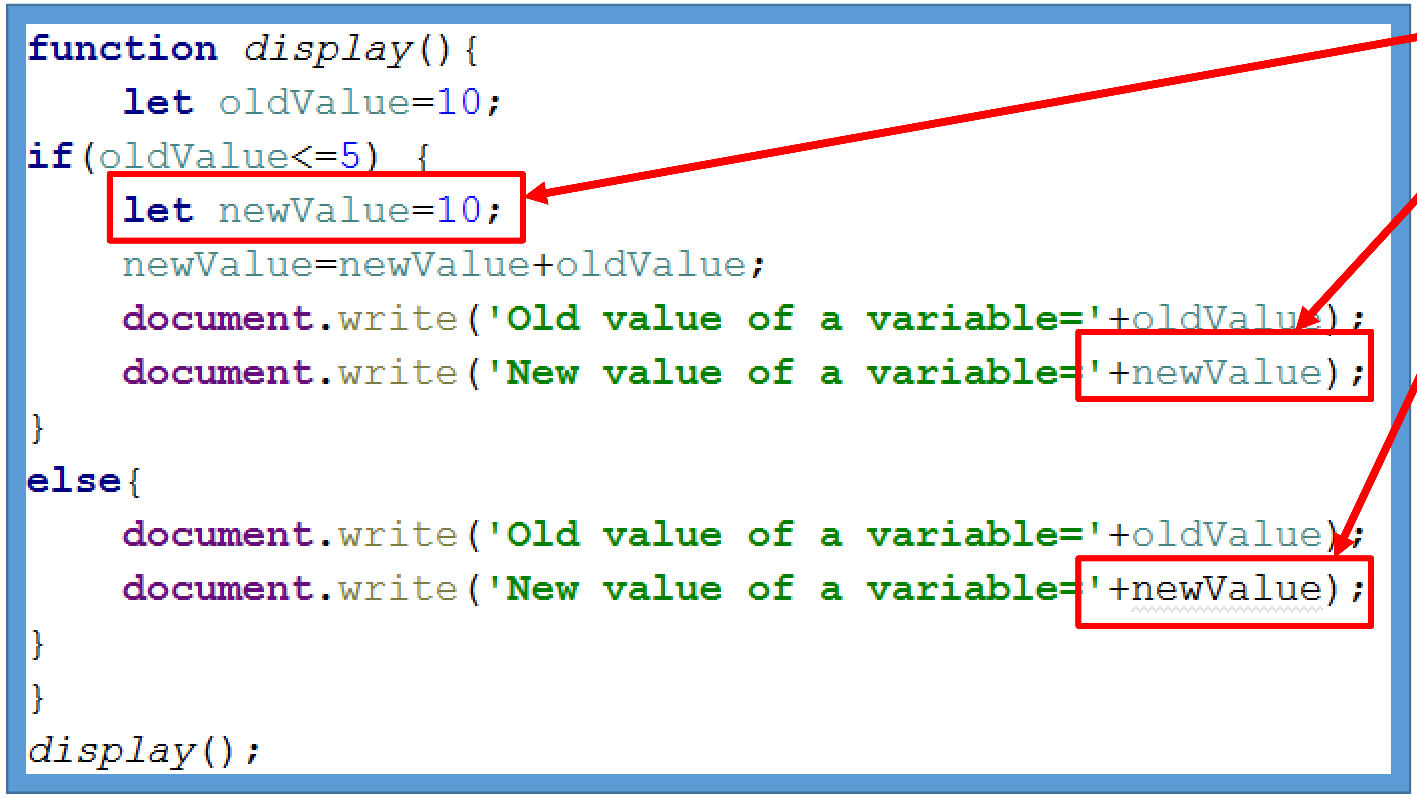
```
function display(){
    let oldValue=10;
if(oldValue<=5) {
    let newValue=10;
    newValue=newValue+oldValue;
    document.write('Old value of a variable='+oldValue);
    document.write('New value of a variable='+newValue);
}
else{
    document.write('Old value of a variable='+oldValue);
    document.write('New value of a variable='+newValue);
}
}
display();
```

oldValue is declared inside the function, it is accessible inside if block and also inside else block

The scope of a variable oldValue is within the function

Demo - let

index.js



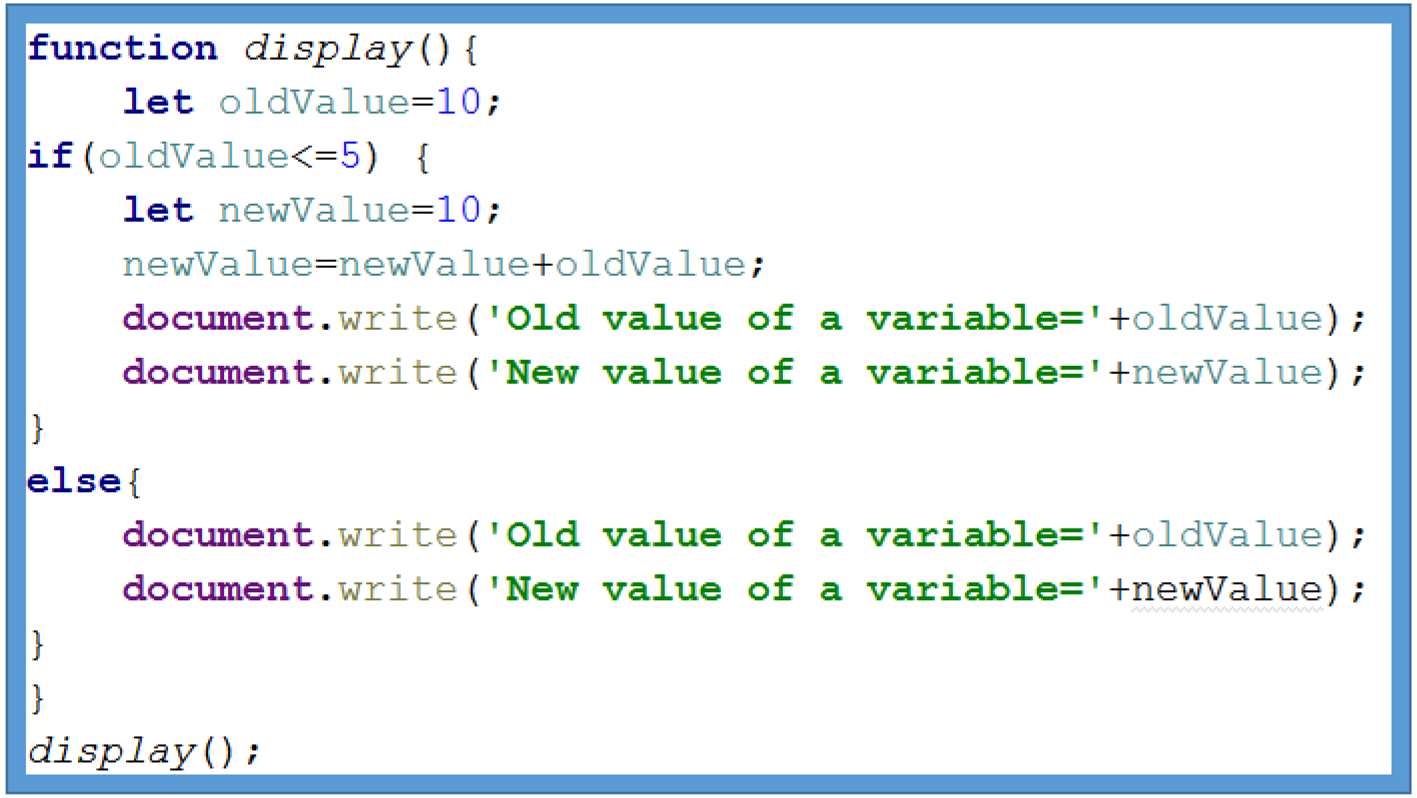
```
function display(){
    let oldValue=10;
if(oldValue<=5) {
    let newValue=10;
    newValue=newValue+oldValue;
    document.write('Old value of a variable='+oldValue);
    document.write('New value of a variable='+newValue);
}
else{
    document.write('Old value of a variable='+oldValue);
    document.write('New value of a variable='+newValue);
}
}
display();
```

newValue is declared inside the if block, it is accessible inside if block but it cannot be accessed inside else block

The scope of a variable newValue is within the if block only

Demo - let

index.js



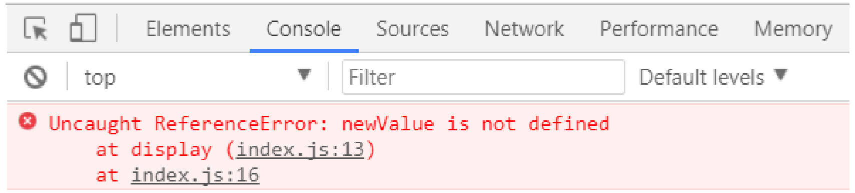
```
function display(){
    let oldValue=10;
if(oldValue<=5) {
    let newValue=10;
    newValue=newValue+oldValue;
    document.write('Old value of a variable='+oldValue);
    document.write('New value of a variable='+newValue);
}
else{
    document.write('Old value of a variable='+oldValue);
    document.write('New value of a variable='+newValue);
}
}
display();
```

Output

```
Old value of a variable=10
```

Hello World from HTML Page

Error on browser console



```
Uncaught ReferenceError: newValue is not defined
    at display (index.js:13)
    at index.js:16
```

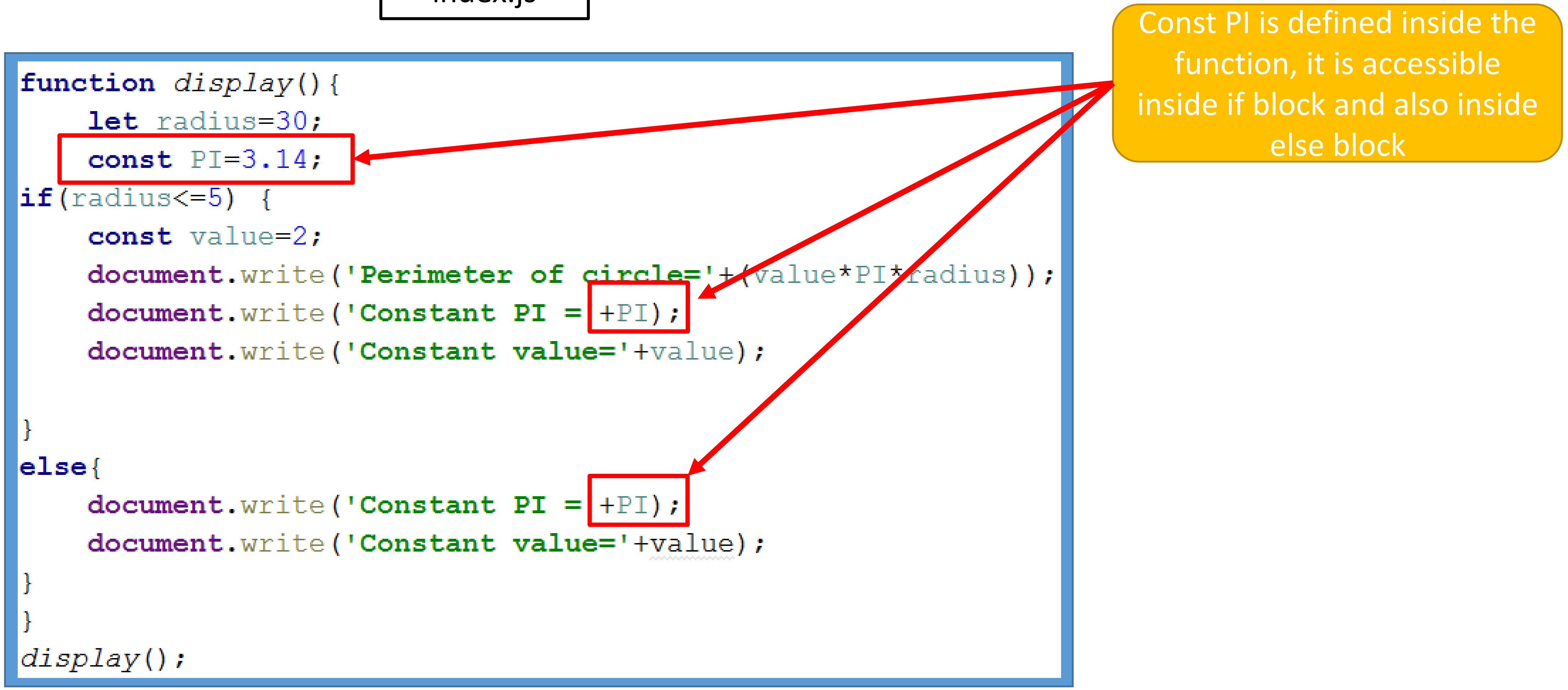
Demo - const

index.js

```
function display(){
    let radius=30;
    const PI=3.14;
if(radius<=5) {
    const value=2;
    document.write('Perimeter of circle='+(value*PI*radius));
    document.write('Constant PI ='+PI);
    document.write('Constant value='+value);

}
else{
    document.write('Constant PI ='+PI);
    document.write('Constant value='+value);
}
}
display();
```

Demo - const

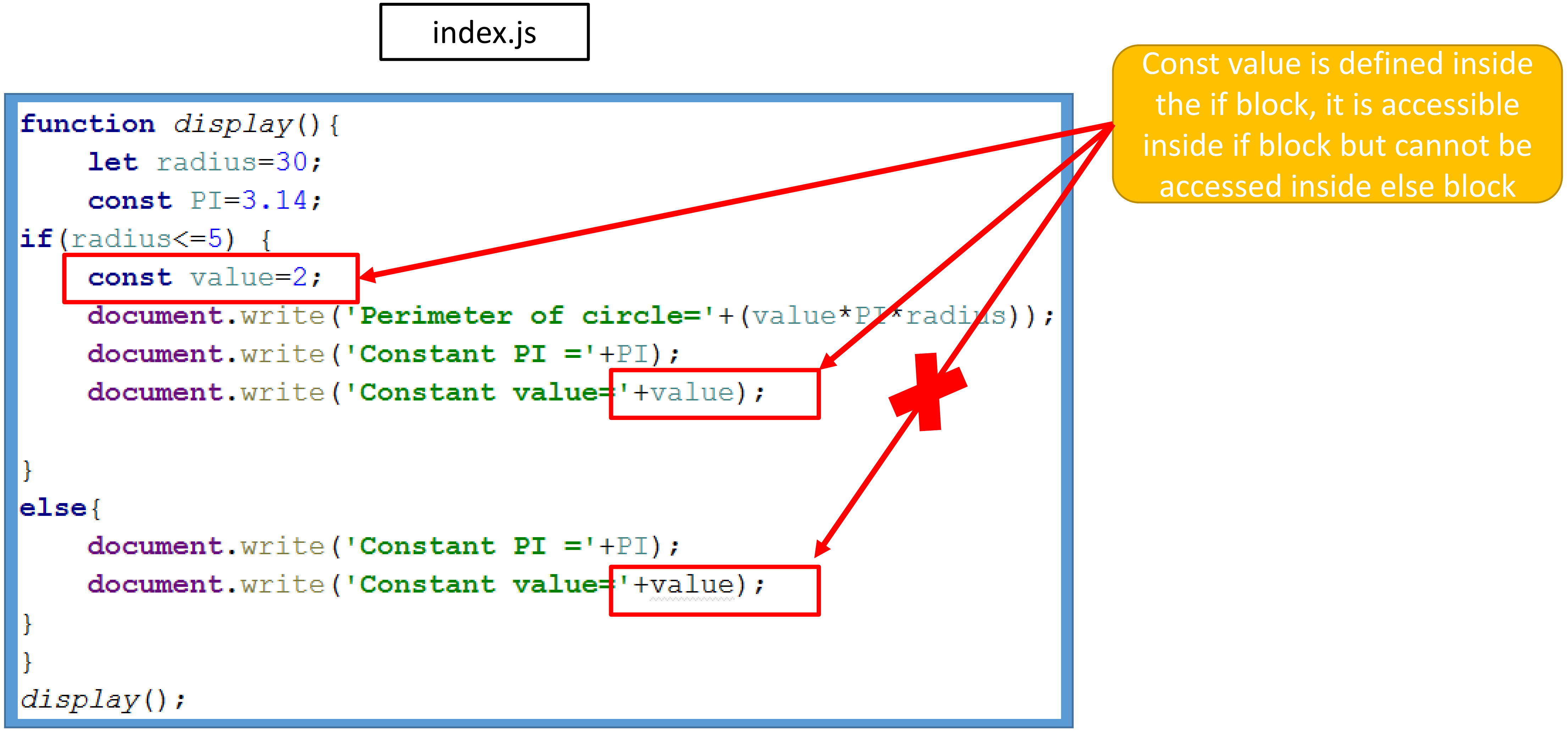
index.js



```
function display(){
    let radius=30;
    const PI=3.14;
if(radius<=5) {
    const value=2;
    document.write('Perimeter of circle='+(value*PI*radius));
    document.write('Constant PI = '+PI);
    document.write('Constant value='+value);

}
else{
    document.write('Constant PI = '+PI);
    document.write('Constant value='+value);
}
}
display();
```

Const PI is defined inside the function, it is accessible inside if block and also inside else block

Demo - const



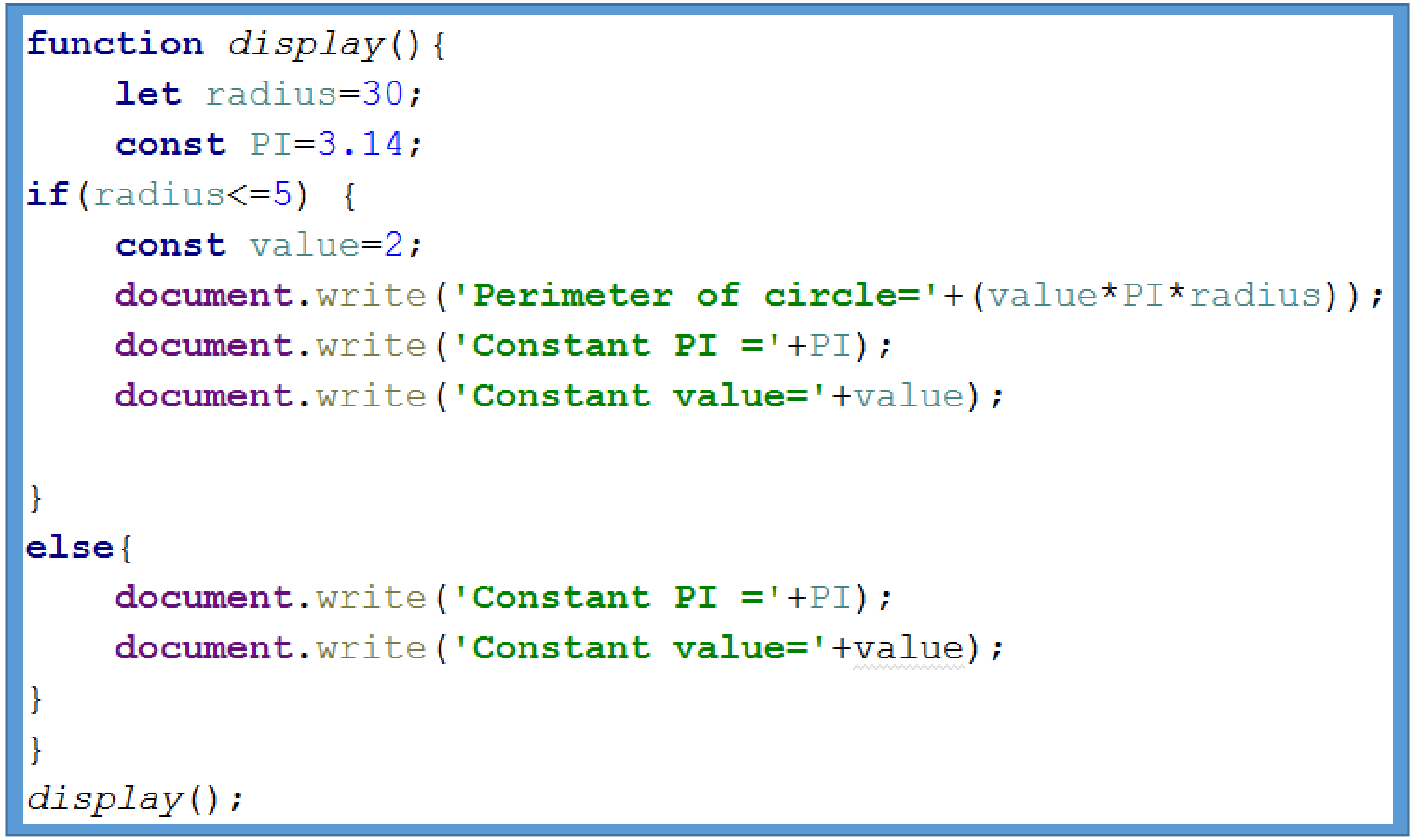
index.js

```
function display(){
    let radius=30;
    const PI=3.14;
if(radius<=5) {
    const value=2;
    document.write('Perimeter of circle='+(value*PI*radius));
    document.write('Constant PI ='+PI);
    document.write('Constant value='+value);

}
else{
    document.write('Constant PI ='+PI);
    document.write('Constant value='+value);
}
}
display();
```

Const value is defined inside the if block, it is accessible inside if block but cannot be accessed inside else block

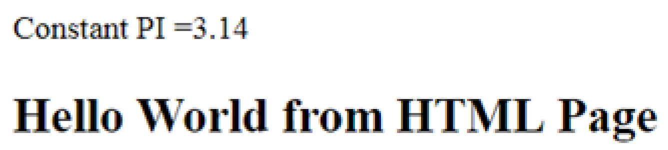
Demo - const

index.js



```
function display(){
    let radius=30;
    const PI=3.14;
if(radius<=5) {
    const value=2;
    document.write('Perimeter of circle='+(value*PI*radius));
    document.write('Constant PI ='+PI);
    document.write('Constant value='+value);

}
else{
    document.write('Constant PI ='+PI);
    document.write('Constant value='+value);
}
}
display();
```

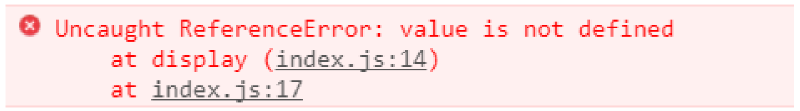
Output



Constant PI =3.14

Hello World from HTML Page

Error on browser console



```
Uncaught ReferenceError: value is not defined
    at display (index.js:14)
    at index.js:17
```

accenture

ES6 New Feature – Class and Inheritance

Class in ES6 can be created using “class” keyword

Class definition can include constructor and methods

Class definition in ES6 cannot have data properties directly in the class body, however these can be inside constructor

Syntax

```
class Classname
 {
            constructor() {
            data properties;
            }
            method() {
            }
   }
```

Instantiation

```
let obj=new Classname();
```

accenture

Demo - class

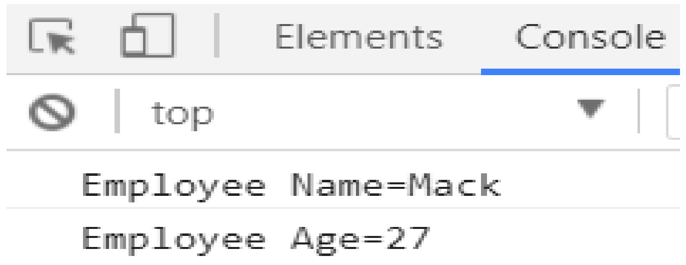
index.js

```
class Employee{
    constructor(name,age){
        this.name=name;
        this.age=age;
    }

    displayEmployee(){
       console.log('Employee Name='+this.name);
       console.log('Employee Age='+this.age);
    }
}

let obj=new Employee('Mack',27);
obj.displayEmployee();
```

Output

Hello World from HTML Page

Output on browser console



```
Employee Name=Mack
Employee Age=27
```

ES6 New Feature – Class and Inheritance

Inheritance in ES6 can be achieved using “extends” keyword

Inheritance is a concept of creating new class using existing class definition

The main class is referred as “parent or super” class and the newly created class is referred as “child or sub” class

The sub class can inherit data and methods from the super class

The sub class cannot inherit constructor from the super class

Whenever there is need to invoke parent class members, this could be achieved using “super” keyword

Syntax

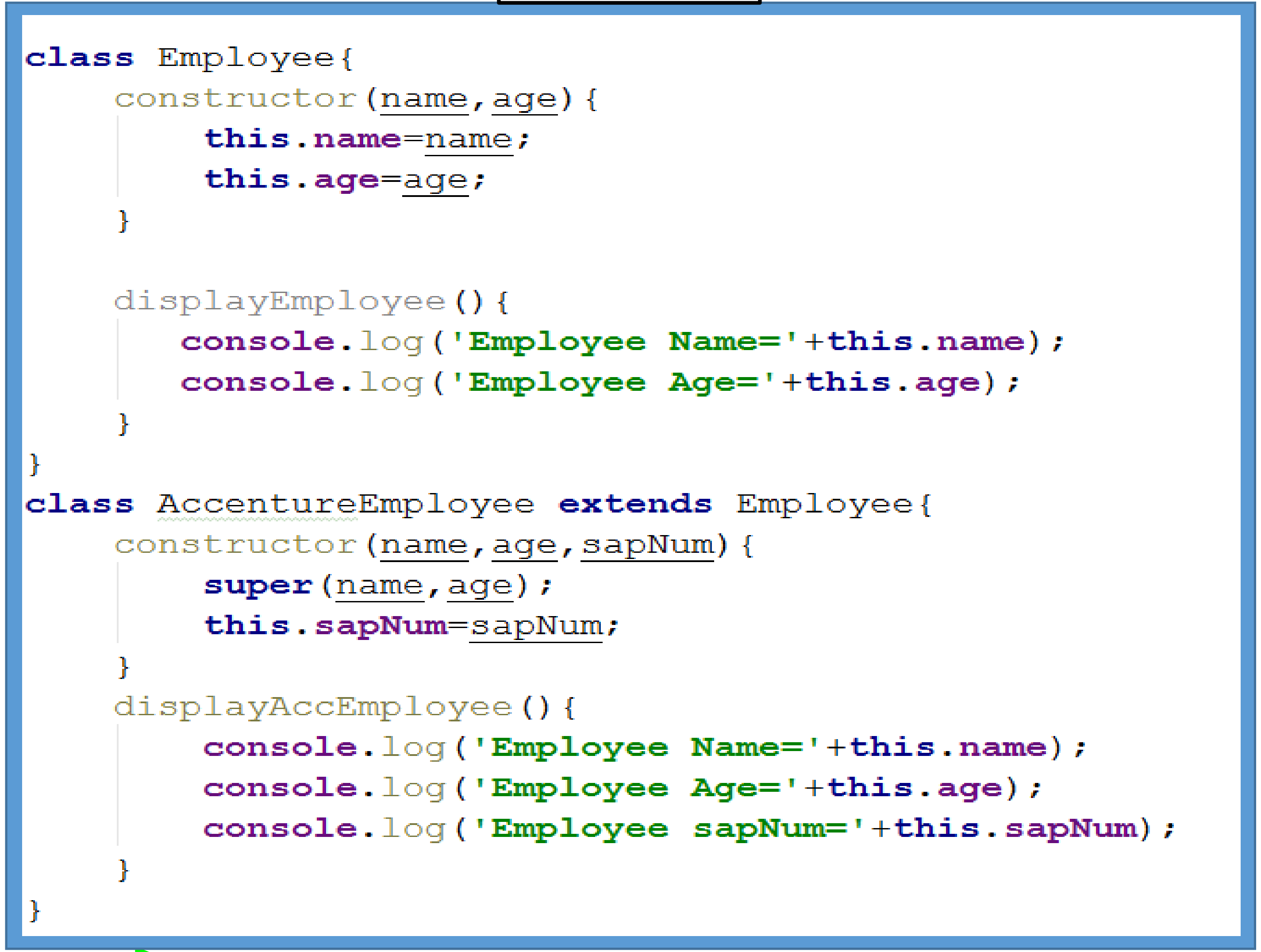
```
class Parentclassname
  {
                constructor() {
                data properties;
                }
                method() {
                }
   }
class Childclassname extends Parentclassname{
constructor() {
                data properties;
                }
                method() {
                }
}
```

Instantiation

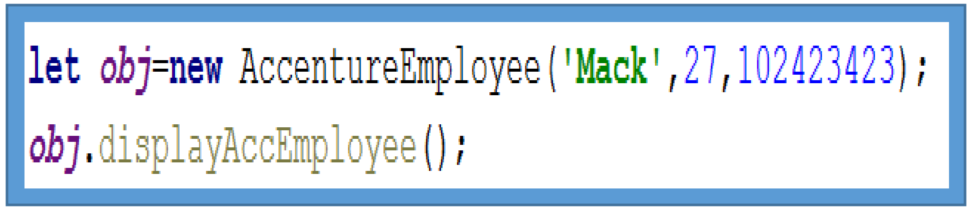
```
let obj=new Childclassname();
```

Demo – class inheritance

index.js



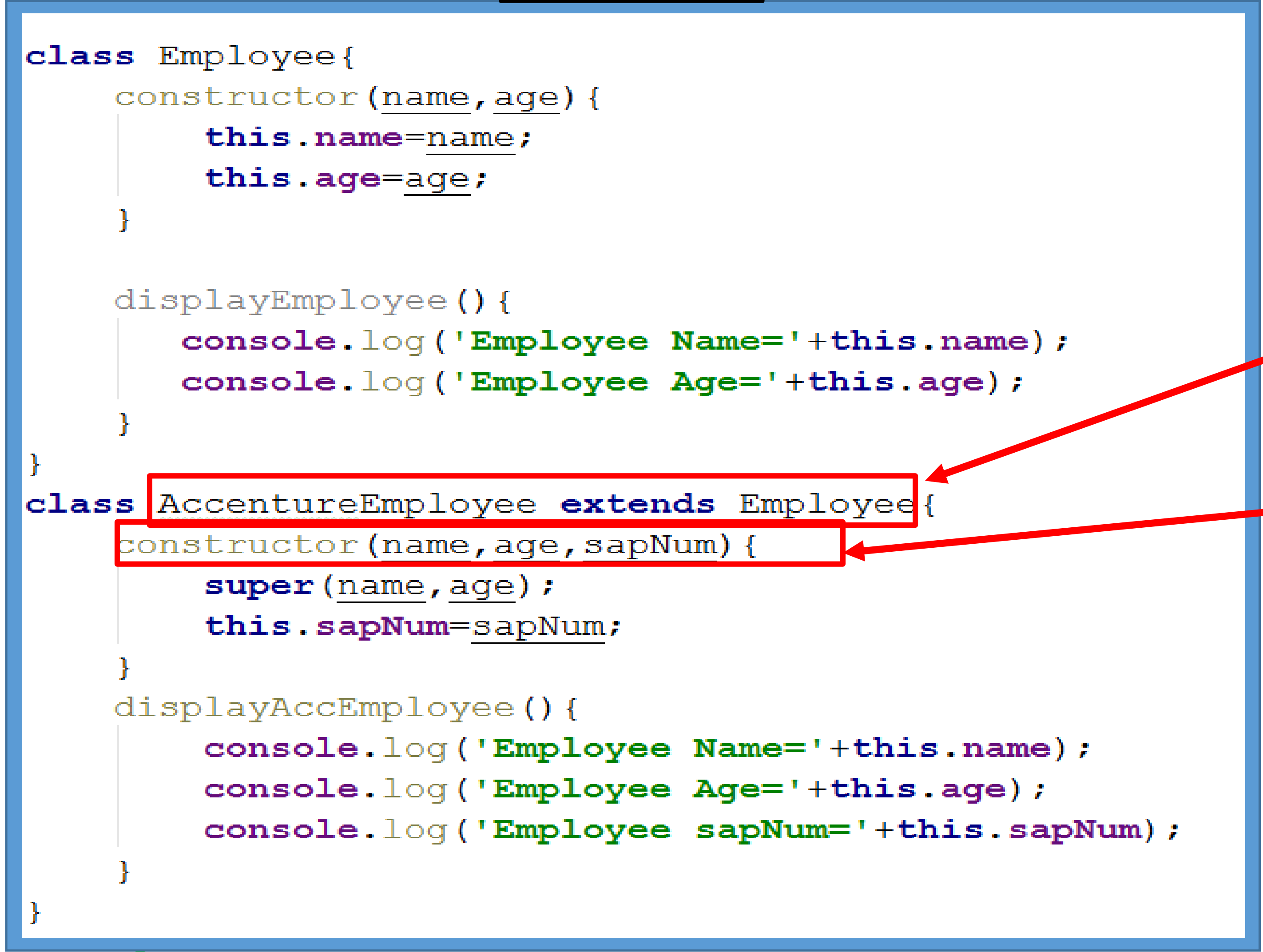
```
class Employee{
    constructor(name,age){
        this.name=name;
        this.age=age;
    }

    displayEmployee(){
      console.log('Employee Name='+this.name);
      console.log('Employee Age='+this.age);
    }
}
class AccentureEmployee extends Employee{
    constructor(name,age,sapNum){
        super(name,age);
        this.sapNum=sapNum;
    }
    displayAccEmployee(){
        console.log('Employee Name='+this.name);
        console.log('Employee Age='+this.age);
        console.log('Employee sapNum='+this.sapNum);
    }
}
```



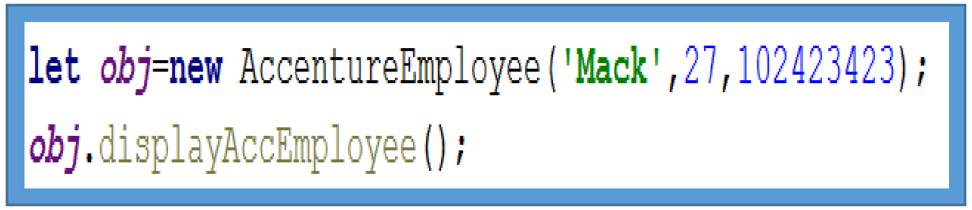
```
let obj=new AccentureEmployee('Mack',27,102423423);
obj.displayAccEmployee();
```

Demo – class inheritance

index.js



```
class Employee{
    constructor(name,age){
        this.name=name;
        this.age=age;
    }

    displayEmployee(){
      console.log('Employee Name='+this.name);
      console.log('Employee Age='+this.age);
    }
}
class AccentureEmployee extends Employee{
    constructor(name,age,sapNum){
        super(name,age);
        this.sapNum=sapNum;
    }
    displayAccEmployee(){
        console.log('Employee Name='+this.name);
        console.log('Employee Age='+this.age);
        console.log('Employee sapNum='+this.sapNum);
    }
}
```



```
let obj=new AccentureEmployee('Mack',27,102423423);
obj.displayAccEmployee();
```

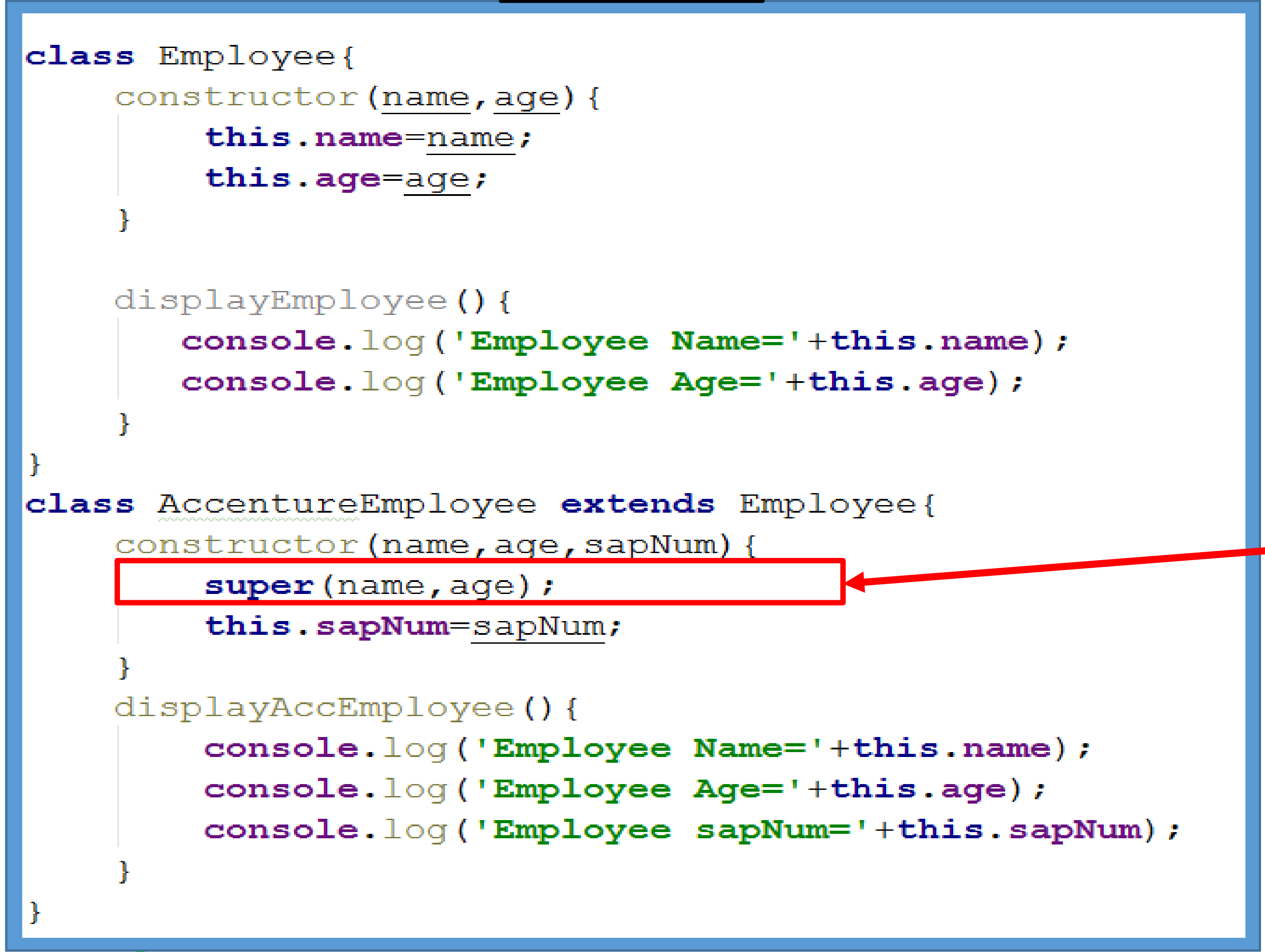
extends keyword used to perform inheritance, here AccentureEmployee class is inherited by Employee class

child class constructor

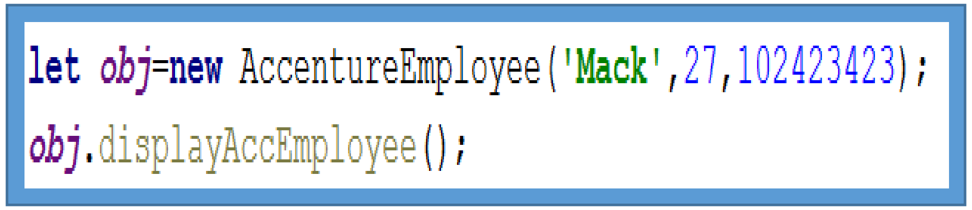
accenture

Demo – class inheritance

index.js



```
class Employee{
    constructor(name,age){
        this.name=name;
        this.age=age;
    }

    displayEmployee(){
       console.log('Employee Name='+this.name);
       console.log('Employee Age='+this.age);
    }
}
class AccentureEmployee extends Employee{
    constructor(name,age,sapNum){
        super(name,age);
        this.sapNum=sapNum;
    }
    displayAccEmployee(){
        console.log('Employee Name='+this.name);
        console.log('Employee Age='+this.age);
        console.log('Employee sapNum='+this.sapNum);
    }
}
```



```
let obj=new AccentureEmployee('Mack',27,102423423);
obj.displayAccEmployee();
```

super keyword is used here to invoke base class constructor

accenture

Demo – class inheritance

index.js

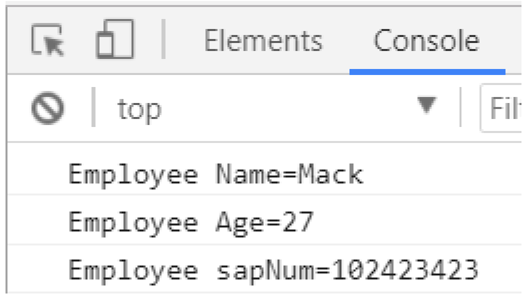
```
class Employee{
    constructor(name,age){
        this.name=name;
        this.age=age;
    }

    displayEmployee(){
      console.log('Employee Name='+this.name);
      console.log('Employee Age='+this.age);
    }
}
class AccentureEmployee extends Employee{
    constructor(name,age,sapNum){
        super(name,age);
        this.sapNum=sapNum;
    }
    displayAccEmployee(){
        console.log('Employee Name='+this.name);
        console.log('Employee Age='+this.age);
        console.log('Employee sapNum='+this.sapNum);
    }
}
```

```
let obj=new AccentureEmployee('Mack',27,102423423);
obj.displayAccEmployee();
```

Output

Hello World from HTML Page

Output on browser console



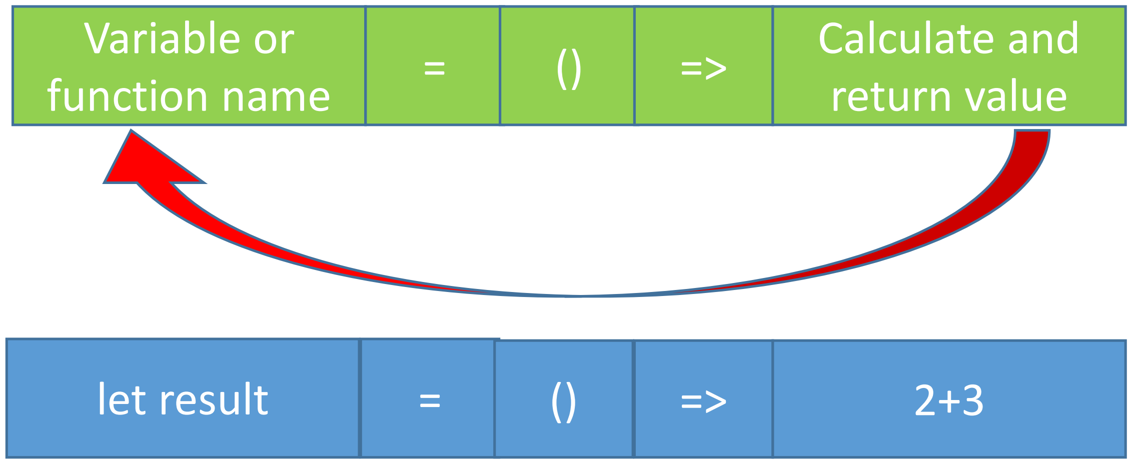
```
Employee Name=Mack
Employee Age=27
Employee sapNum=102423423
```

ES6 New Feature – Arrow Function

ES6 provide this type of arrow "=>" which is also known as "Fat" Arrow

Benefit of using fat arrow is, it will shorten the code with respect to function body

Syntax



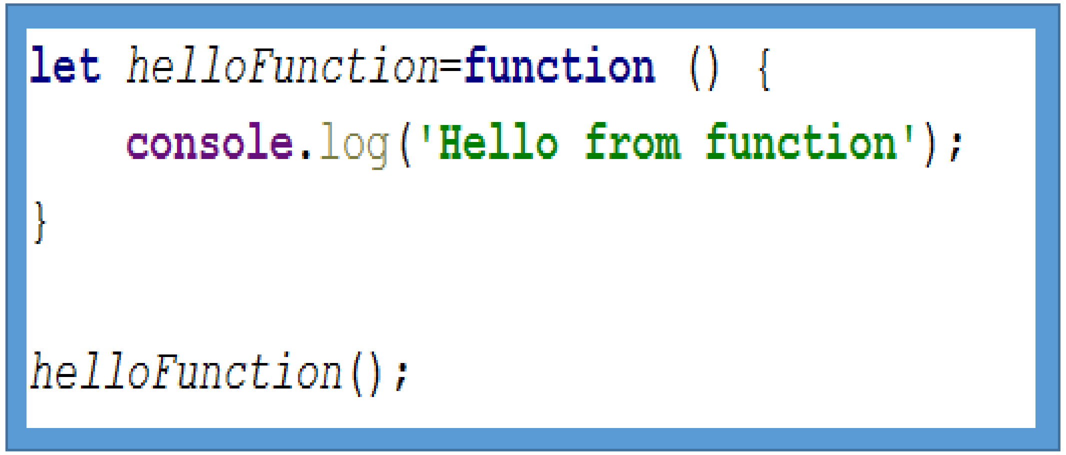
Variable or function name	=	()	=>	Calculate and return value

let result	=	()	=>	2+3

Right side of fat arrow is used to either calculate or return some value and left side of fat arrow will capture the returned or calculated value

Demo – Arrow Function or Fat Arrow

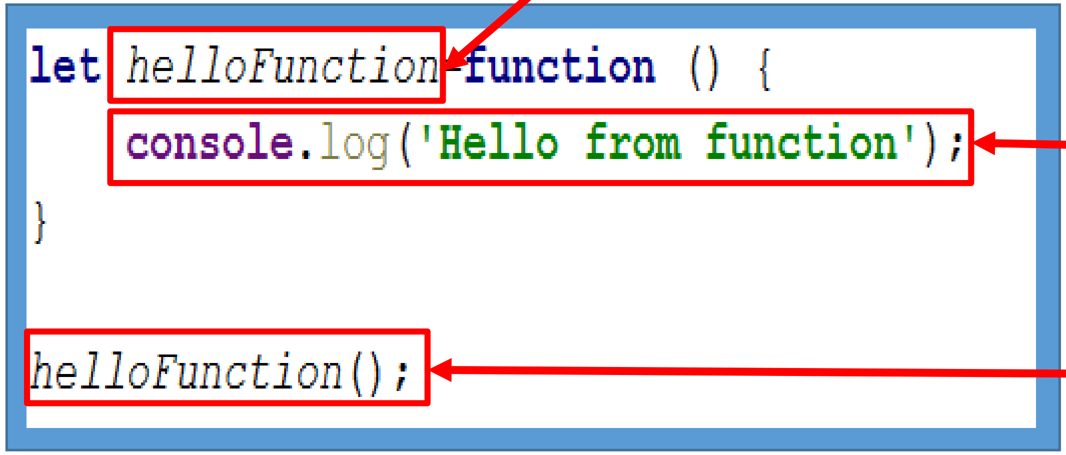
index.js



```
let helloFunction=function () {
    console.log('Hello from function');
}

helloFunction();
```

Demo – Arrow Function or Fat Arrow

index.js



```
let helloFunction=function () {
    console.log('Hello from function');
}

helloFunction();
```

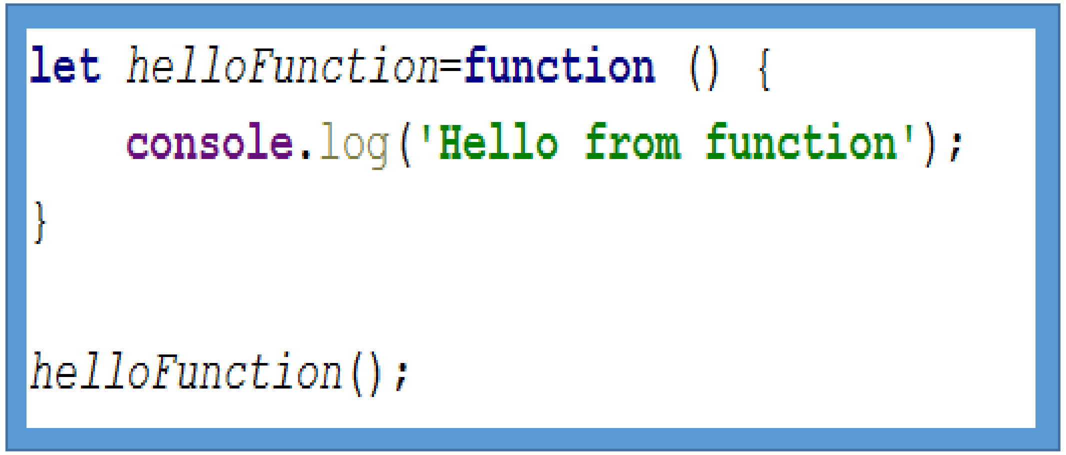
Variable Name or it can also be considered as function name

Function body which may have printing statement or it may return some value

Invoking function

Demo – Arrow Function or Fat Arrow

index.js

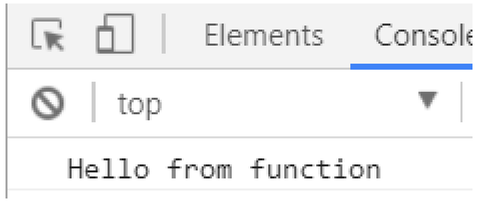


```
let helloFunction=function () {
    console.log('Hello from function');
}

helloFunction();
```

Output

Hello World from HTML Page

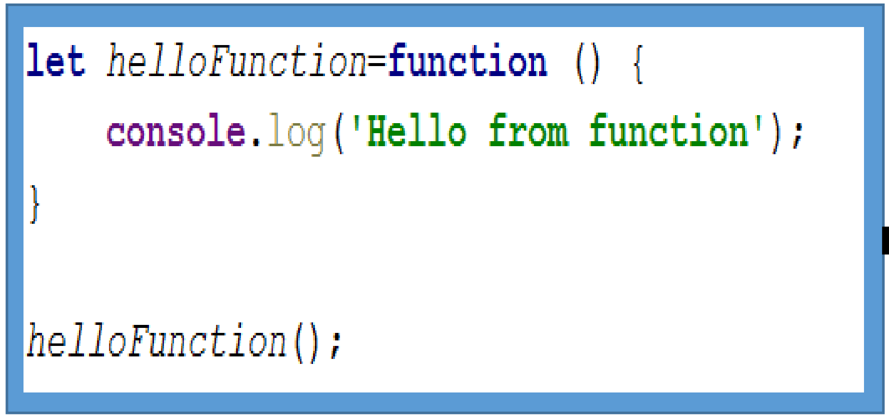
Output on browser console



```
Hello from function
```

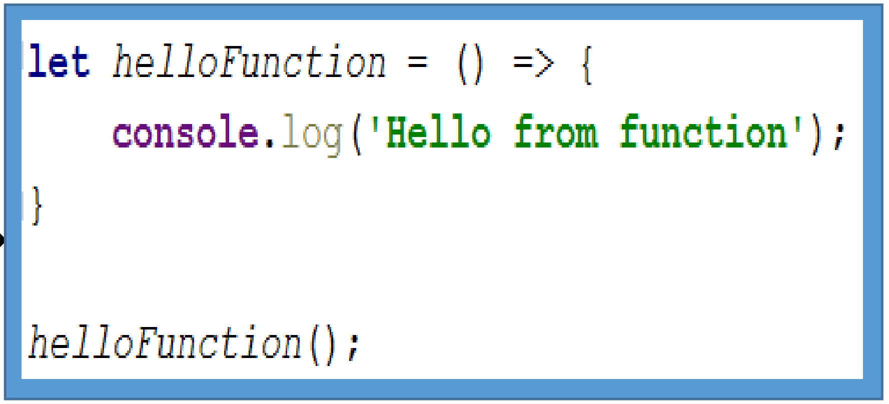
accenture

Demo – Arrow Function or Fat Arrow



```
let helloFunction=function () {
    console.log('Hello from function');
}

helloFunction();
```

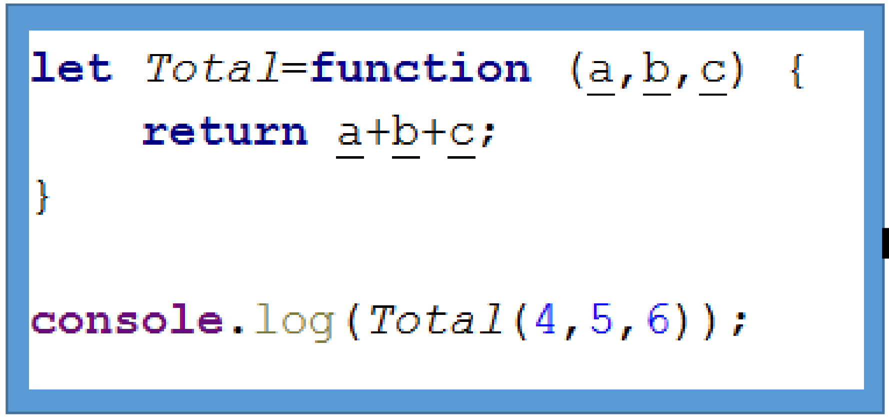
Same function when created using Arrow function



```
let helloFunction = () => {
    console.log('Hello from function');
}

helloFunction();
```

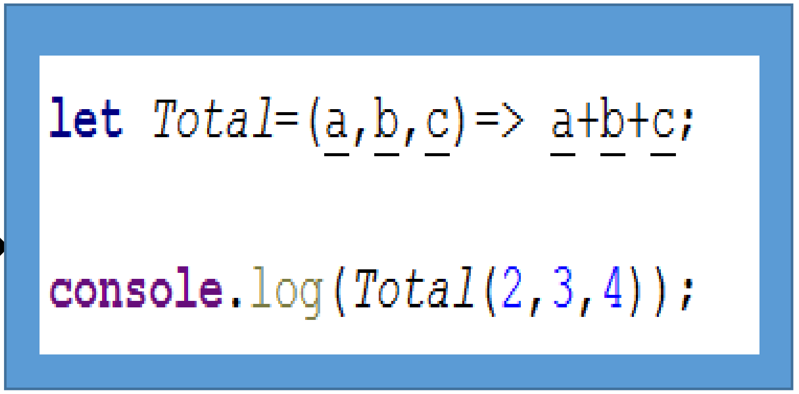
Function keyword is not used

Demo – Arrow Function With Arguments



```
let Total=function (a,b,c) {
    return a+b+c;
}

console.log(Total(4,5,6));
```

Same function when created using Arrow function



```
let Total=(a,b,c)=> a+b+c;


console.log(Total(2,3,4));
```

Function keyword is not used

return keyword is not used

{ } are not used

=> Is used which will shorten the code

accenture

ES6 New Feature – Default value in Function

Function can have parameters, but some parameters may have default value assigned

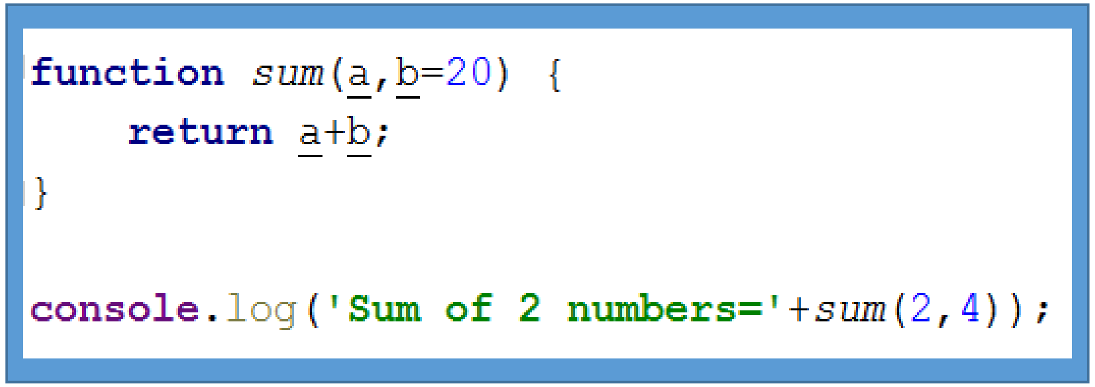
While invoking the function, if a user do not pass value to that parameter then default will be considered

Syntax

```
function function_name(arg1,arg2=defaultValue)
```

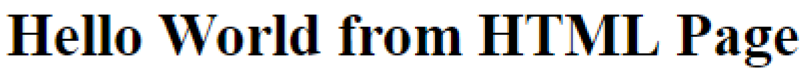
Demo – Default value in function

index.js



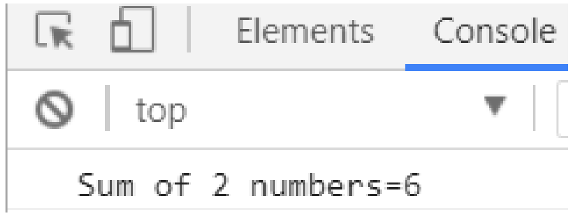
```
function sum(a,b=20) {
    return a+b;
}

console.log('Sum of 2 numbers='+sum(2,4));
```

Output



Hello World from HTML Page

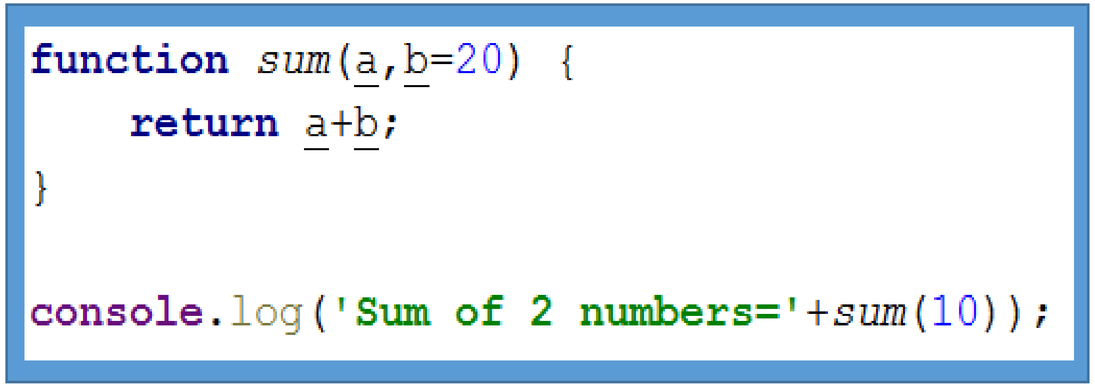
Output on browser console



```
Sum of 2 numbers=6
```

Demo – Default value in function

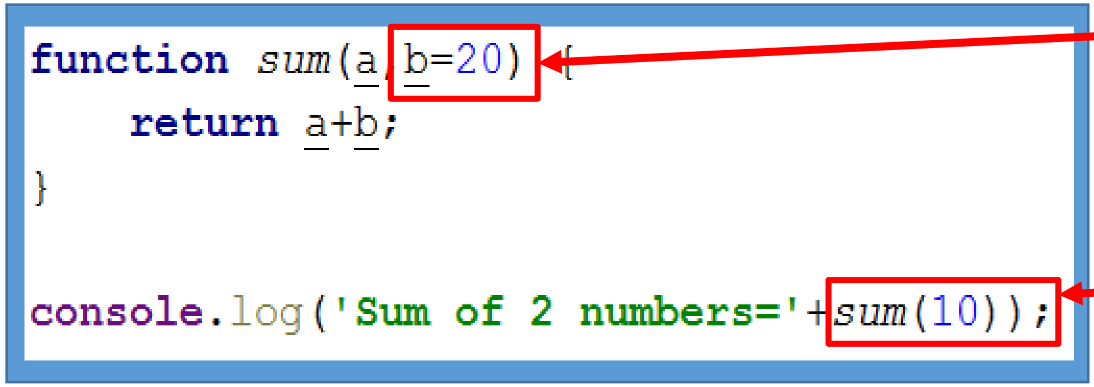
index.js


```
function sum(a,b=20) {
    return a+b;
}

console.log('Sum of 2 numbers='+sum(10));
```

accenture

Demo – Default value in function

index.js



```
function sum(a,b=20) {
    return a+b;
}

console.log('Sum of 2 numbers='+sum(10));
```

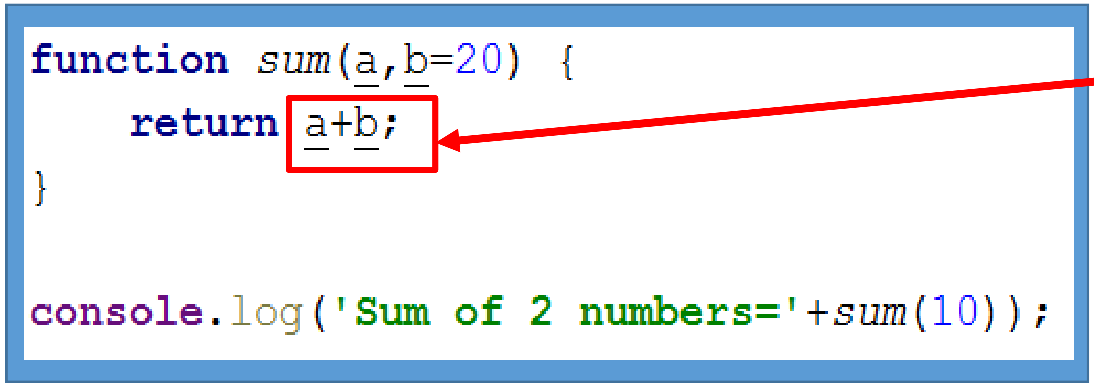
Default value 20 is assigned to the parameter b

Function is invoked and passed value for parameter a but not for parameter b

accenture

Demo – Default value in function

index.js



```
function sum(a,b=20) {
    return a+b;
}

console.log('Sum of 2 numbers='+sum(10));
```

Here a value is 10 and b value is 20. hence the output will be displayed as 30

accenture

Demo – Default value in function

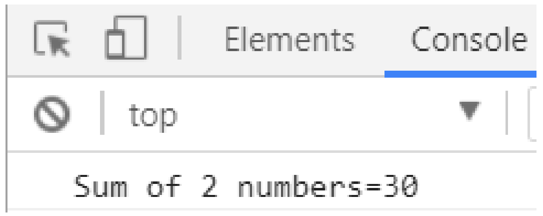
index.js

```
function sum(a,b=20) {
    return a+b;
}

console.log('Sum of 2 numbers='+sum(10));
```

Output

Hello World from HTML Page

Output on browser console



```
Sum of 2 numbers=30
```

ES6 New Feature – Destructuring

Destructuring is a way of unpacking the values. It could be from array or returned value from a function etc.

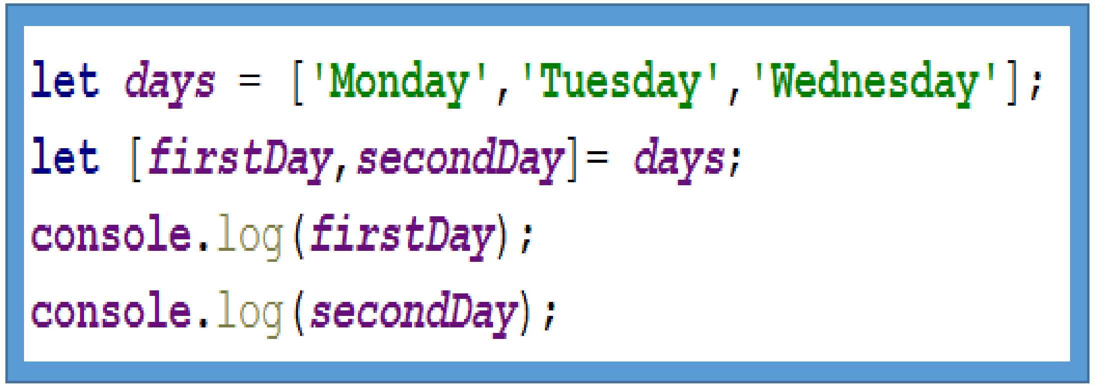
Unpacked elements could be assigned to other variables

<u>Syntax</u>

```
var Variable_name= array[elements];
```

Demo – Destructuring

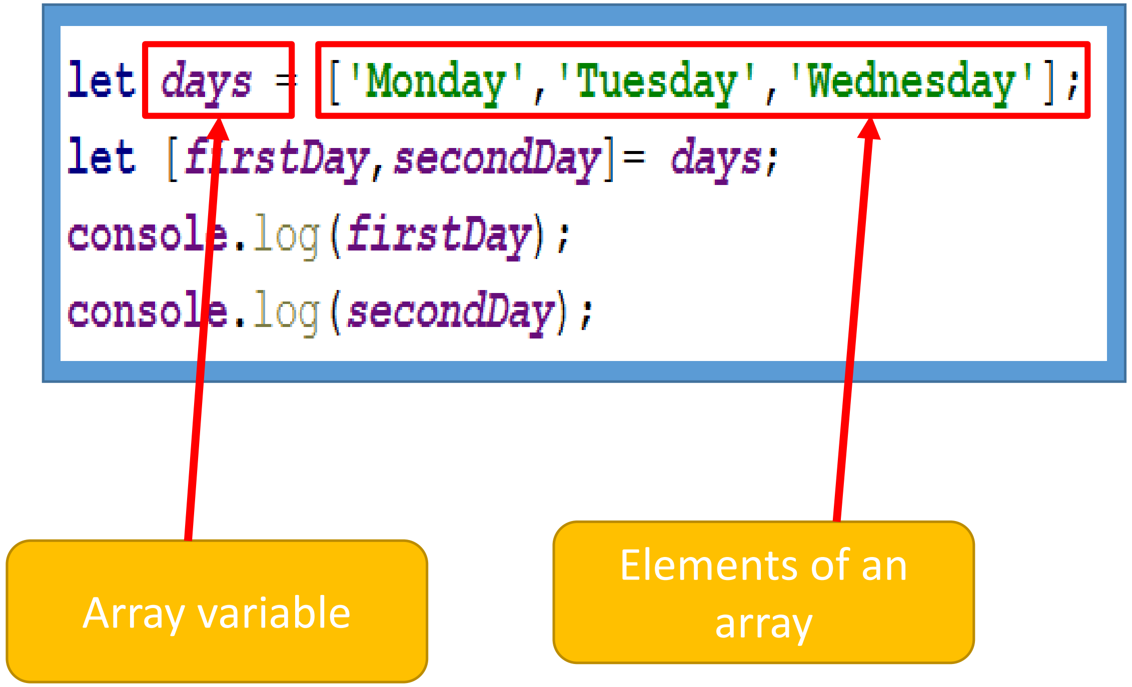
index.js



```
let days = ['Monday','Tuesday','Wednesday'];
let [firstDay,secondDay]= days;
console.log(firstDay);
console.log(secondDay);
```

Demo – Destructuring

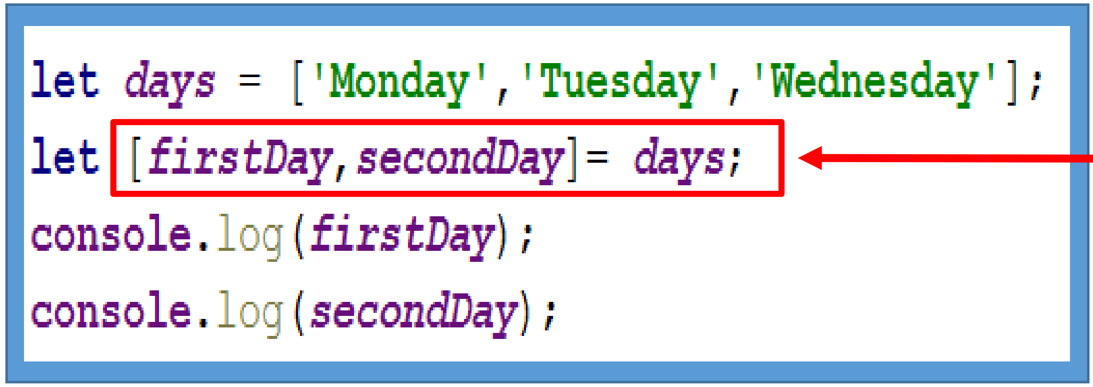
index.js



```
let days = ['Monday','Tuesday','Wednesday'];
let [firstDay,secondDay]= days;
console.log(firstDay);
console.log(secondDay);
```

Demo – Destructuring

index.js

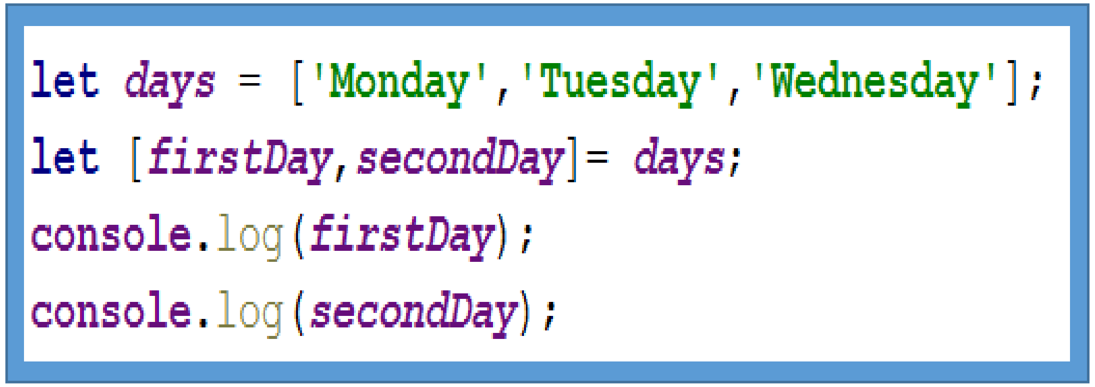


```
let days = ['Monday','Tuesday','Wednesday'];
let [firstDay,secondDay]= days;
console.log(firstDay);
console.log(secondDay);
```

Elements from array(days) are unpacked and assigned to individual variables(firstDay and secondDay). This is known as Destructuring.

Demo – Destructuring

index.js

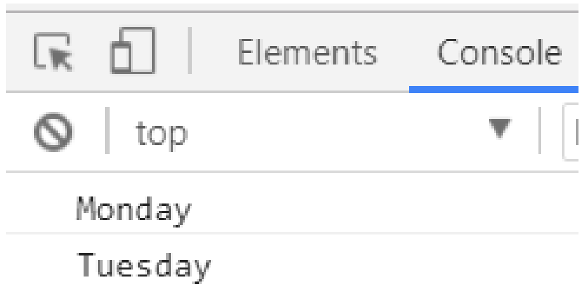


```
let days = ['Monday','Tuesday','Wednesday'];
let [firstDay,secondDay]= days;
console.log(firstDay);
console.log(secondDay);
```

Output

Hello World from HTML Page

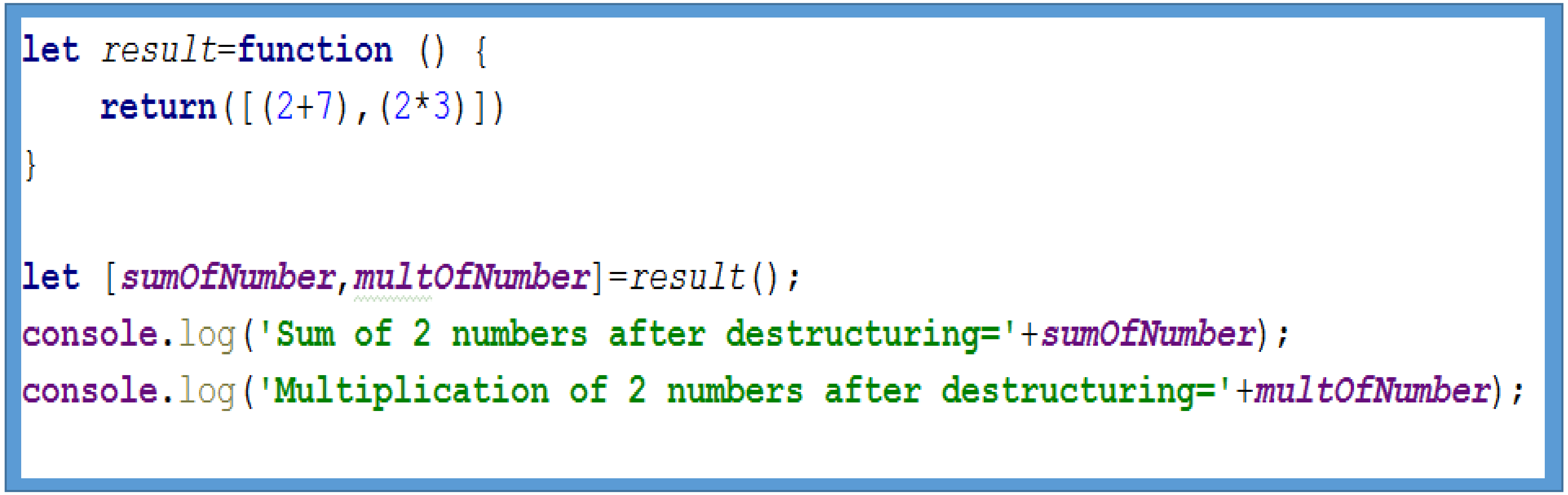
Output on browser console



```
Monday
Tuesday
```

Demo – Destructuring with returned value from function

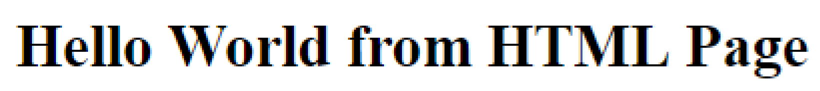
index.js



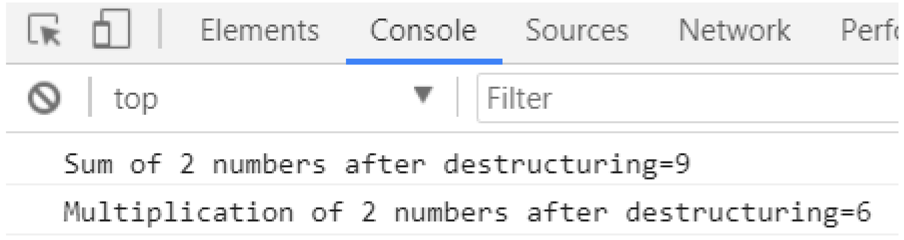
```
let result=function () {
    return([(2+7),(2*3)])
}

let [sumOfNumber,multOfNumber]=result();
console.log('Sum of 2 numbers after destructuring='+sumOfNumber);
console.log('Multiplication of 2 numbers after destructuring='+multOfNumber);
```

Output

Output on browser console



Hello World from HTML Page



```
Sum of 2 numbers after destructuring=9
Multiplication of 2 numbers after destructuring=6
```

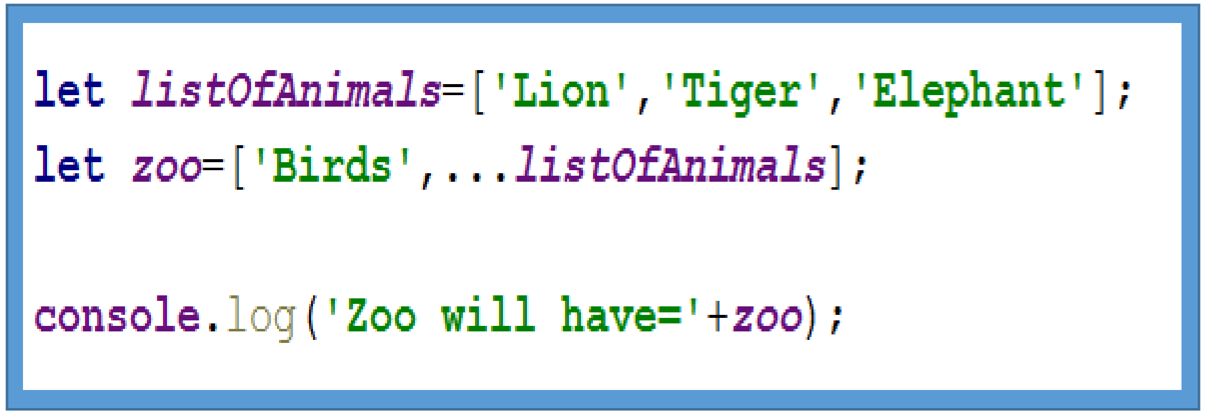
ES6 New Feature – Spread Operator

This operator is used expand or spread the values into different arguments

It can represent zero or more arguments based on the program logic

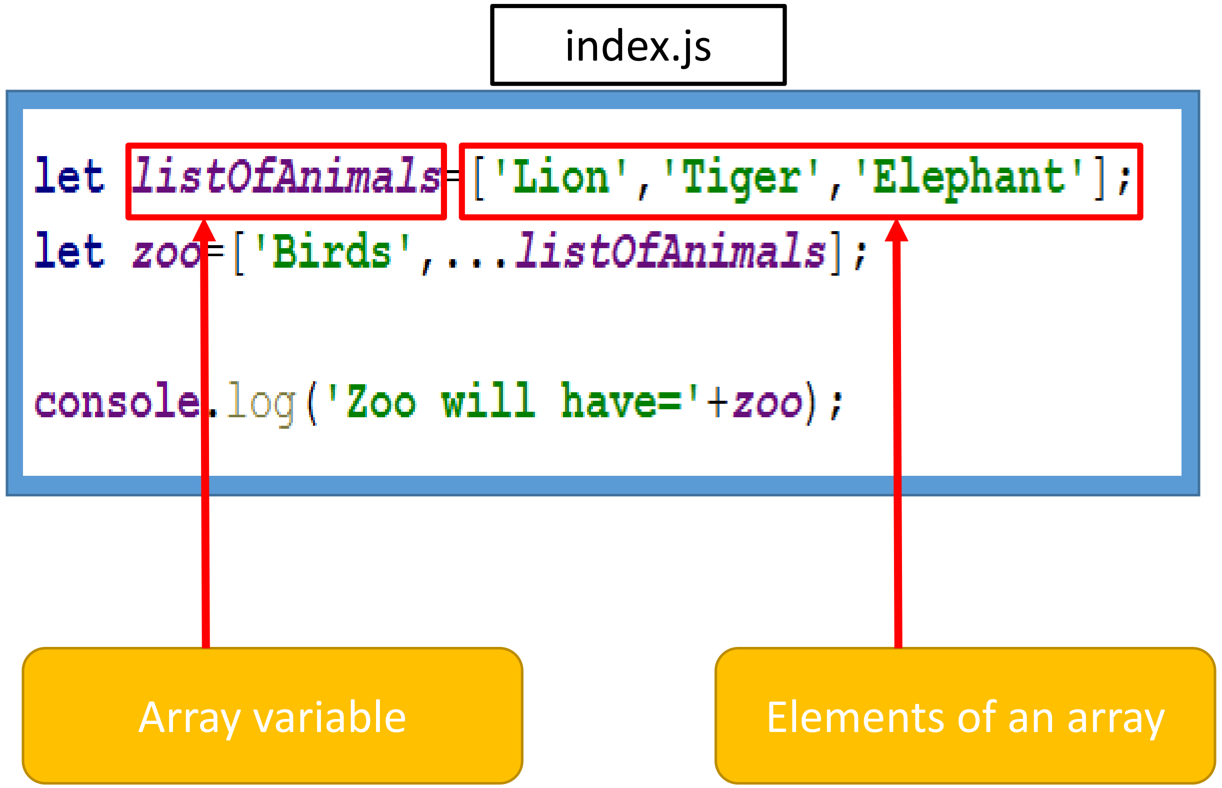
Demo – Spread Operator

index.js



```
let listOfAnimals=['Lion','Tiger','Elephant'];
let zoo=['Birds',...listOfAnimals];

console.log('Zoo will have='+zoo);
```

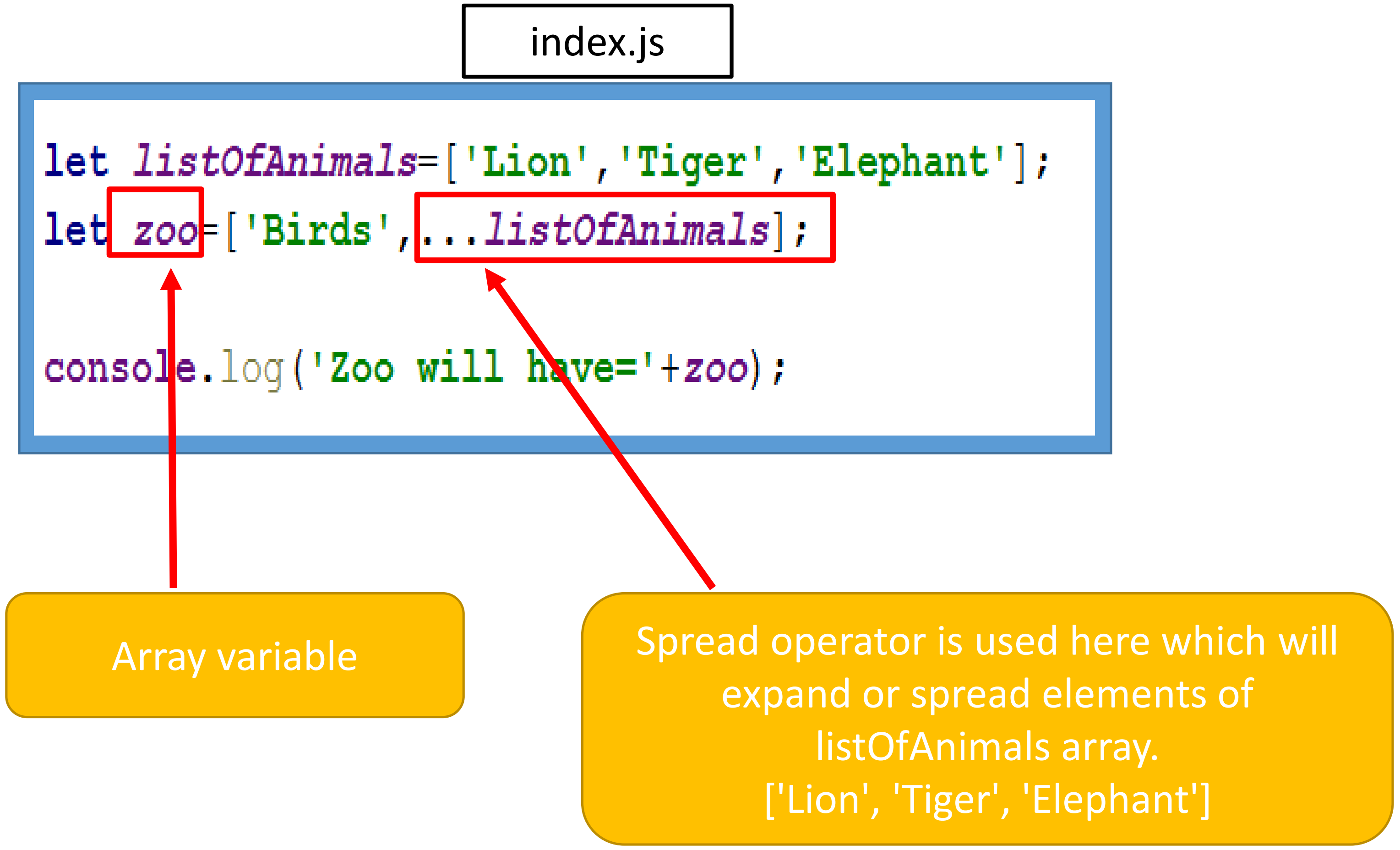
Demo – Spread Operator



index.js

```
let listOfAnimals=['Lion','Tiger','Elephant'];
let zoo=['Birds',...listOfAnimals];

console.log('Zoo will have='+zoo);
```

Demo – Spread Operator

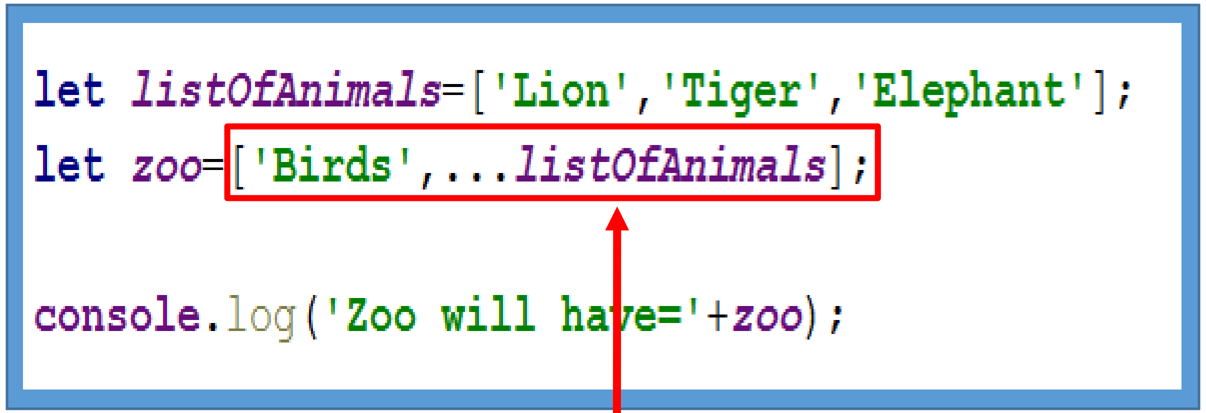

index.js

```
let listOfAnimals=['Lion','Tiger','Elephant'];
let zoo=['Birds',...listOfAnimals];

console.log('Zoo will have='+zoo);
```

Array variable
Spread operator is used here which will expand or spread elements of listOfAnimals array.
['Lion', 'Tiger', 'Elephant']

Demo – Spread Operator

index.js



```
let listOfAnimals=['Lion','Tiger','Elephant'];
let zoo=['Birds',...listOfAnimals];

console.log('Zoo will have='+zoo);
```

Birds element is combined with elements of listOfAnimals array.

Demo – Spread Operator

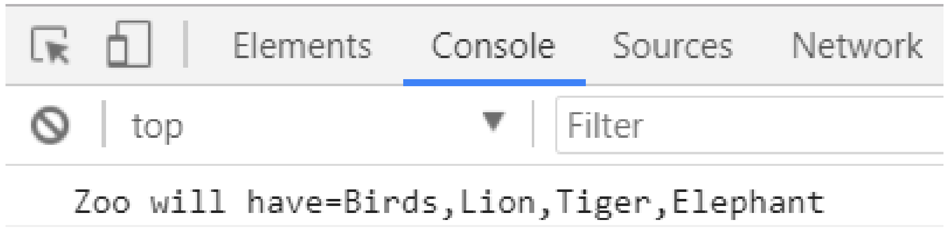
index.js

```
let listOfAnimals=['Lion','Tiger','Elephant'];
let zoo=['Birds',...listOfAnimals];

console.log('Zoo will have='+zoo);
```

Output

Hello World from HTML Page

Output on browser console



```
Zoo will have=Birds,Lion,Tiger,Elephant
```

Demo – Spread Operator

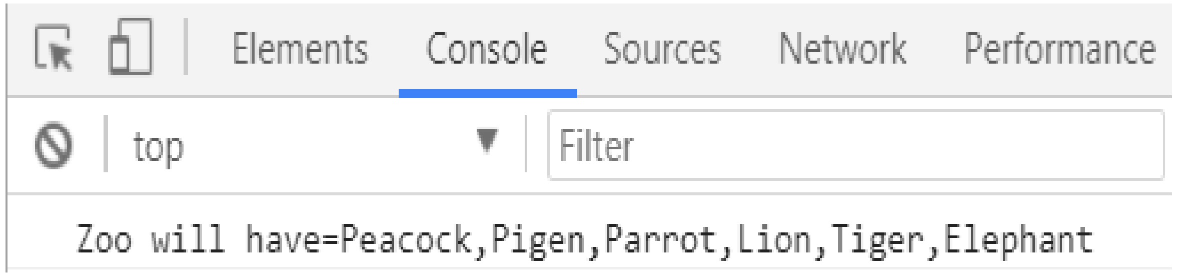
index.js

```
let listOfAnimals=['Lion','Tiger','Elephant'];
let birds=['Peacock','Pigen','Parrot'];
let zoo=[...birds,...listOfAnimals];

console.log('Zoo will have='+zoo);
```

Output

Hello World from HTML Page

Output on browser console



```
Zoo will have=Peacock,Pigen,Parrot,Lion,Tiger,Elephant
```

Demo – Spread Operator In Function Argument

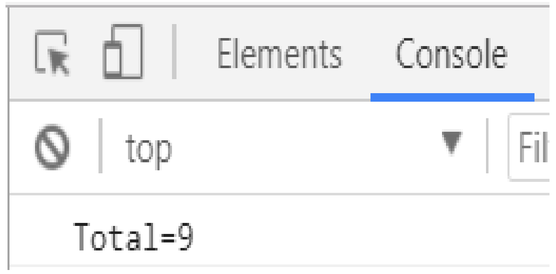
index.js

```
let params=[2,3,4];
function calculateTotal(num1,num2,num3) {
    return num1+num2+num3;
}

console.log('Total='+calculateTotal(...params));
```

Output

Hello World from HTML Page

Output on browser console



```
Total=9
```

MODULE SUMMARY

- Introduction to ES6
- New features of ES6
- Let and const keywords
- Class and inheritance
- Arrow function
- Destructuring
- Spread operator

THANK YOU

accenture