JS

ECMA Script 2015

#JavaScript Notes

ECMA Script 2015
ES6

JS

ES 6

Index

- Let Const
- Template Literals
- Arrow Functions
- Spread Operator
- Rest Parameters

Let,Const

JavaScript let

- let is similar to var but let has scope.
- let is **only accessible in the block** level it is defined.

```
let a = 0;
console.log(a); //0

if (true) {
   let a = 10;
   console.log(a); //10
  }
console.log(a); // 0
```

```
let a = 10;
console.log(a); //10

if(true){
    let a = 90;
    console.log(a); //90
}
console.log(a); //10
```

```
var a = 10;
console.log(a); //10

if(true){
    var a = 90;
    console.log(a); //90
}
console.log(a); //90
```

JavaScript const

- Const is used to assign a constant value to the variable.
- And the value cannot be changed. Its fixed.

```
1    const a = 10;
2    console.log(a); //Print 10
3
4    a = 50; //Error
5
```

JavaScript Template Literals (Template Strings)

JS

Template Literals

- Template literals provide an easy and clean way create multi-line strings and perform string interpolation.
- Now we can embed variables or expressions into a string easily .
- They are enclosed in backticks ``.


```
1    console.log("I Am String"); //Doublt Quote
2    console.log(`I Am String`); // BackTick
```

Template Literals Example

demo.js > ...

```
1    let str = `Template literal in ES6`;
2
3    console.log(str);// Template literal in ES6
4    console.log(str.length); // 23
5    console.log(typeof str);// string
```

Multiline Strings Using Template Literals

- Template literals also make it easy to write multiline strings.

```
// using the + operator
const message1 = 'This is a long message\n' +
'that spans across multiple lines\n' +
'in the code.'

console.log(message1)
```

demo.js > ...

```
1   // Simple multi-line string
2   let a = `Hello
3                   How Are you ?.
4                       Thanks`;
5
6   console.log(a); //Print Data
```

```
1   var a = 10;
2
3   console.log("A Value is \n " + a);
4
5   console.log(`A Value is
6    ${a}`);
```

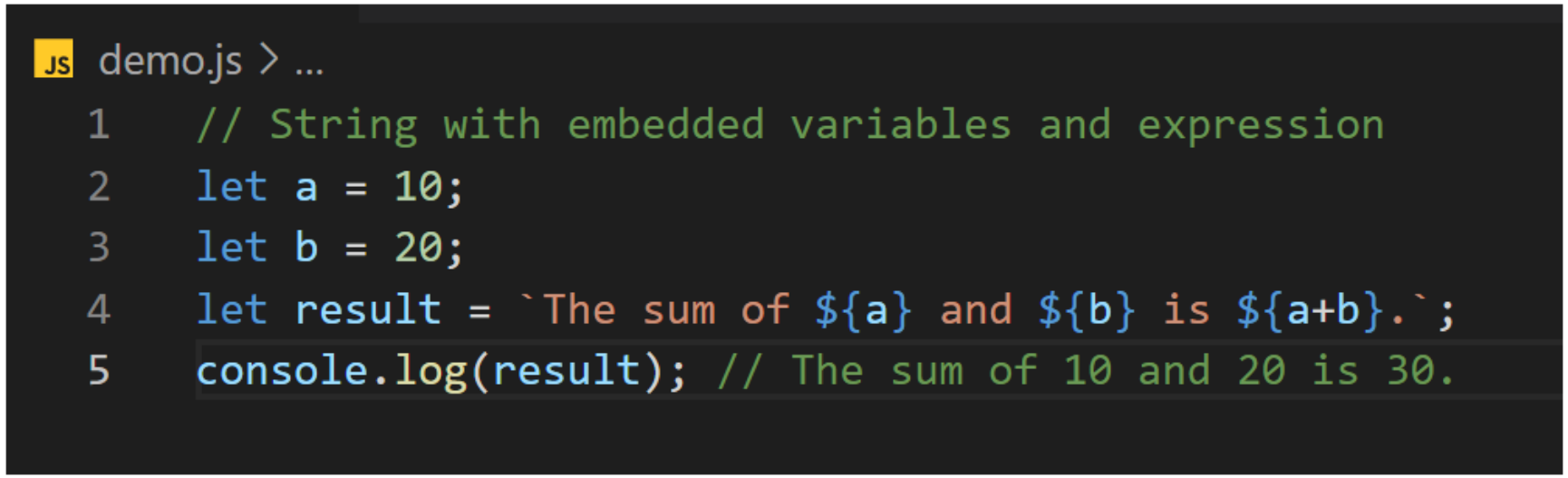
Variable Expression

- Variables or expressions can be placed inside the string using the ${...}

```
1   var a = 10;
2   console.log("A Value is " + a); //Double Quote
3   console.log(`A Value is ${a}`); // BackTick
```

```
const myname = 'Akash';
console.log(`Hello ${myname}!`); // Hello Akash!
```

Sum of 2 Numbers




```
// String with embedded variables and expression
let a = 10;
let b = 20;
      let result = `The sum of ${a} and ${b} is ${a+b}.`;
      console.log(result); // The sum of 10 and 20 is 30.
```

Escape Character

- In the earlier versions of JavaScript, you would use a single quote '' or a double quote "" for strings.
- To use the same quotations inside the string, you can use the escape character \.

```
1   console.log("Happy Mother\"s Day!");
2   console.log('Happy Mother\'s Day!');
3   console.log(`Happy Mother's Day`);
```

Arrow

JavaScript Arrow Function

- In the ES6 version, you can use arrow functions to create function expressions.
- Use the (...args) => expression; to define an arrow function.
- Use the (...args) => { statements } to define an arrow function that has multiple statements.

```
// Function expression
let x = function(x, y) {
   return x * y;
}

//Arrow Function
let x = (x, y) => x * y;
```

demo.js > ...

```
1   // Function expression
2   let x = function(x, y) {
3       return x * y;
4   }
5
6   //Arrow Function
7   let x = (x, y) => x * y;
8
```

Example 1: Arrow Function with No Argument

- If a function doesn't take any argument, then you should use empty parentheses.

demo.js > ...

```
1   // Function expression
2   let msg = () => console.log("Hello World")
3   msg(); // Hello World
```

```
1   function msg()
2   {
3       console.log("Hello world");
4   }
5   msg();
```

// Function expression
let msg = () => console.log("Hello World")
msg(); // Hello World

Example 2: Arrow Function with One Argument

- If a function has only one argument, you can omit the parentheses.

demo.js > ...

```
1   // Function expression
2   let msg = x => console.log(x)
3   msg("Hello World"); // Hello World
4
```

```
// Function expression
let msg = x => console.log(x)
msg("Hello World"); // Hello World
```

```
function msg(x)
{
    console.log(x);
}
msg("Hello world");
```

Arrow Function with Argument and Return

demo.js > ...

```
1   let add = (x, y) => x + y;
2
3   console.log(add(10, 20)); // 30;
```

```
1   function add(x,y)
2   {
3       return x+y;
4   }
5   console.log(add(10,20));
```

let add = (x, y) => x + y;

console.log(add(10, 20)); // 30;

Example 3: Arrow Function as an Expression

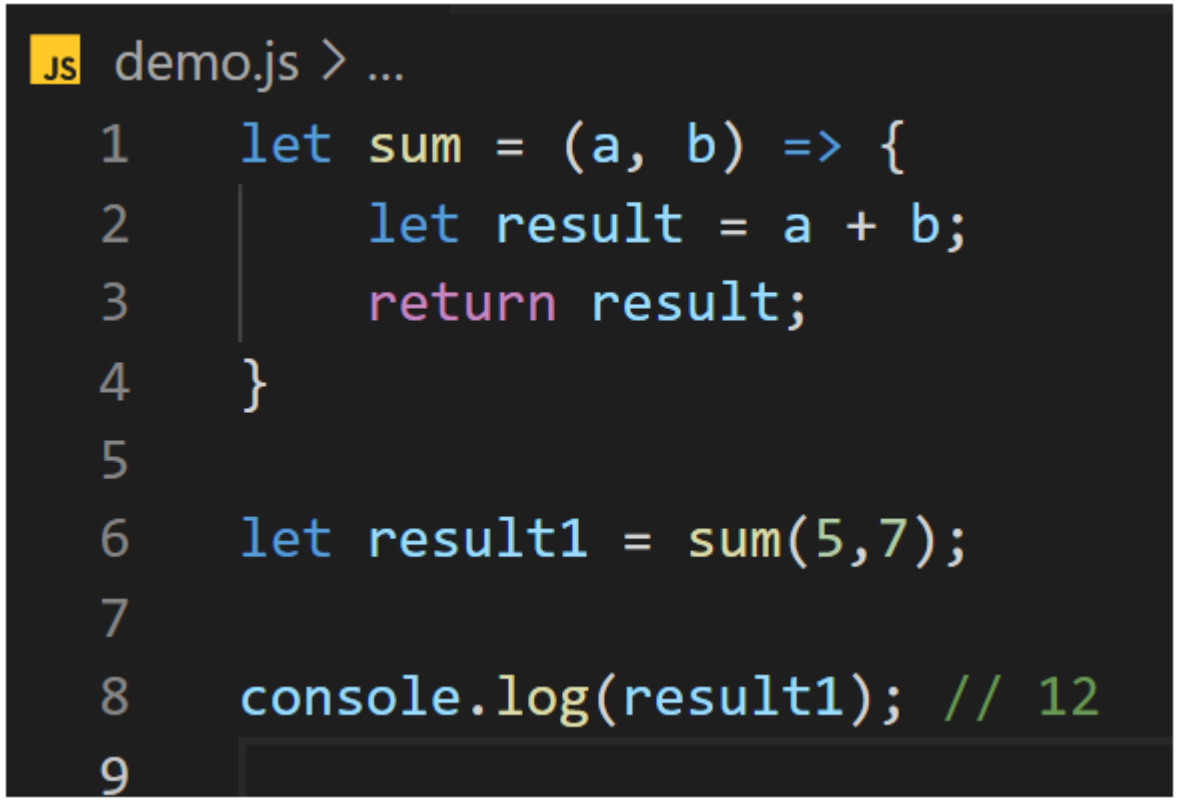
- You can also dynamically create a function and use it as an expression.

```
let age = 5;

let welcome = (age < 18) ?
  () => console.log('Baby') :
  () => console.log('Adult');

welcome(); // Baby
```

Example 4: Multiline Arrow Functions

- If a function body has multiple statements, you need to put them inside curly brackets {}.

```
let sum = (a, b) => {
    let result = a + b;
    return result;
}

let result1 = sum(5,7);

console.log(result1); // 12
```



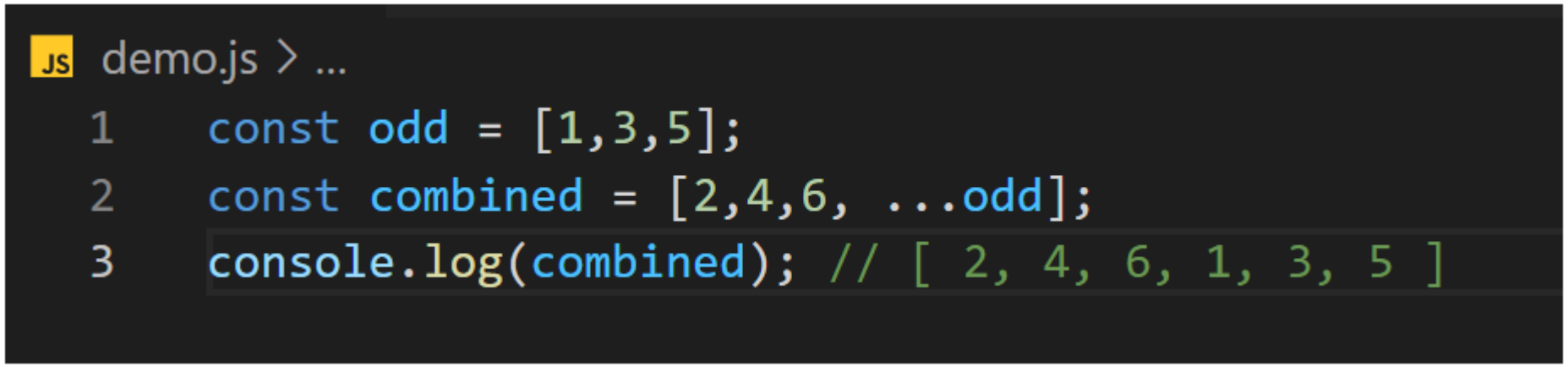
```js
//1 Simple Function
let msg = () => console.log("Function Called");
msg();

//2 Parameter
let greeting = (x) => console.log("Hi My Name is " + x);
greeting("Akash");

//3 Parameter with Return Value
let sum = (x,y) => x+y;
var ans = sum(10,20);
console.log("Sum is " + ans);

//4 Ternary Condition
let age = 5;
let welcome = (age < 18) ?        //Condition
    () => console.log('Baby') :   //True
    () => console.log('Adult');   //False
welcome(); // Baby

//5 Retunr Value Multiple Line
let addition = (a, b) => {
    let result = a + b;
    return result;
}
let result1 = addition(5,7);
console.log(result1); // 12
```

```
PS D:\code\jsevening> node .\Aero.js
Function Called
Hi My Name is Akash
Sum is 30
Baby
12
```

Spread operator

Spread … operator

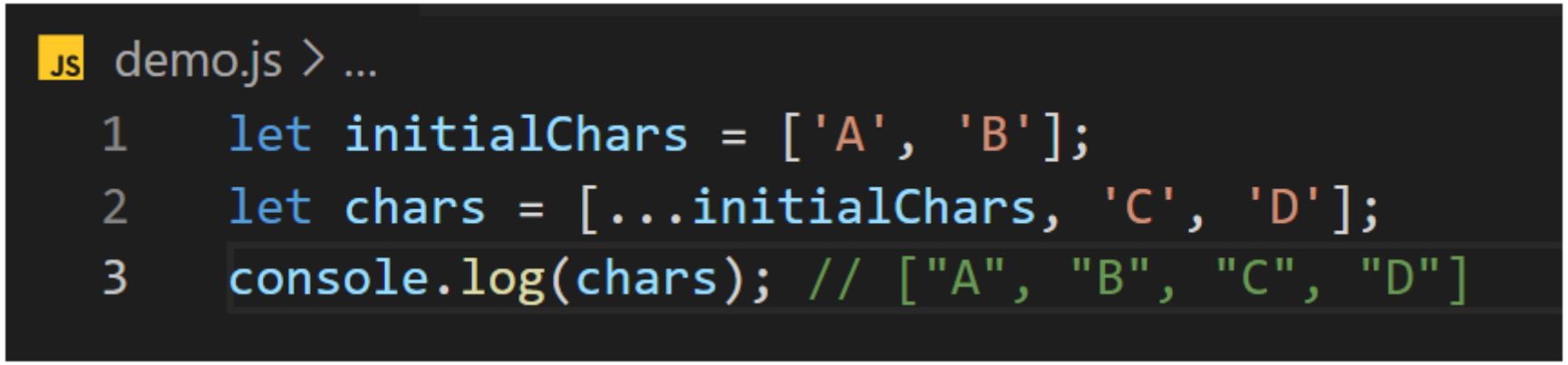
- ES6 provides a new operator called **spread operator that consists of three dots (...)**.
- The spread operator allows you to spread out elements of an iterable object such as **an array, map, or set.**




```
const odd = [1,3,5];
const combined = [2,4,6, ...odd];
console.log(combined); // [ 2, 4, 6, 1, 3, 5 ]
```

1) Constructing array literal

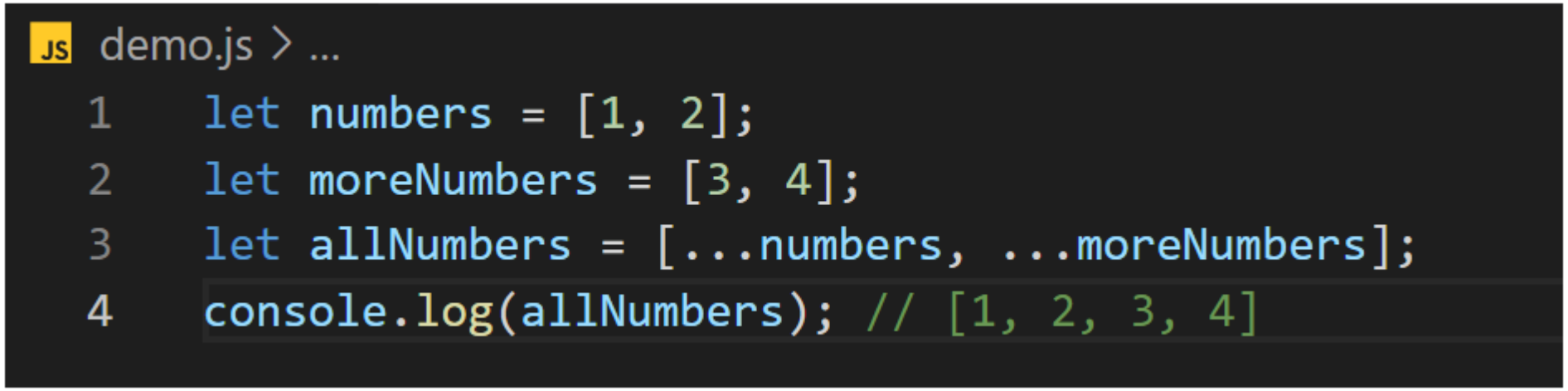
- The spread operator allows you to insert another array into the initialized array when you construct an array using the literal form.




```
let initialChars = ['A', 'B'];
let chars = [...initialChars, 'C', 'D'];
console.log(chars); // ["A", "B", "C", "D"]
```

2) Concatenating arrays

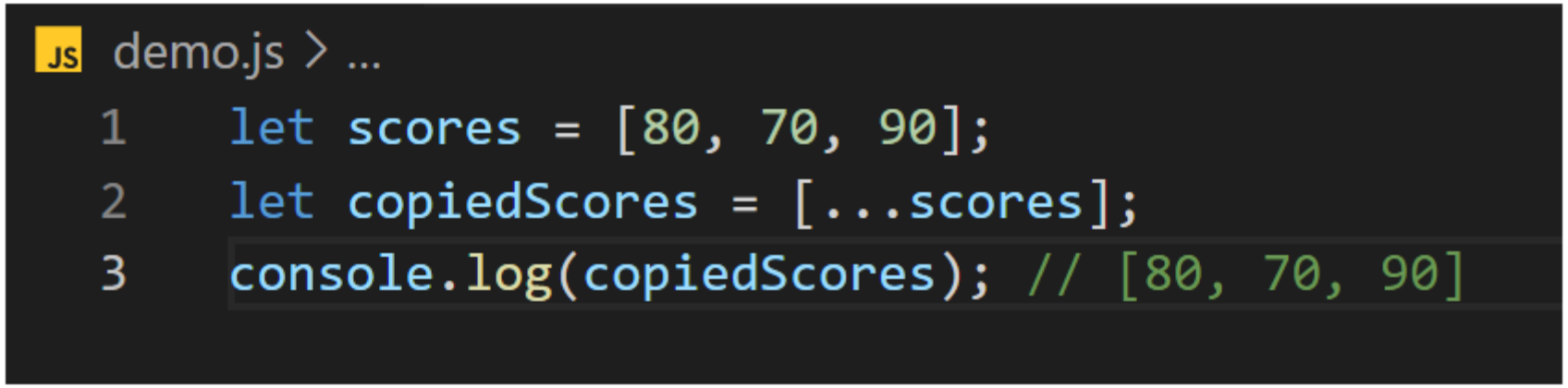
- Also, you can use the spread operator to concatenate two or more arrays:




```
let numbers = [1, 2];
let moreNumbers = [3, 4];
let allNumbers = [...numbers, ...moreNumbers];
console.log(allNumbers); // [1, 2, 3, 4]
```

3) Copying an array

- In addition, you can copy an array instance by using the spread operator:




```
let scores = [80, 70, 90];
let copiedScores = [...scores];
console.log(copiedScores); // [80, 70, 90]
```

Rest Parameter

Rest Parameter

- When the **spread operator is used as a parameter,** it is known as the rest parameter.
- You can also accept multiple arguments in a function call using the rest parameter.

Example

- When a single argument is passed to the func() function, the rest parameter takes only one parameter.
- When three arguments are passed, the rest parameter takes all three parameters.
- Using the rest parameter will pass the arguments as array elements.

demo.js > ...

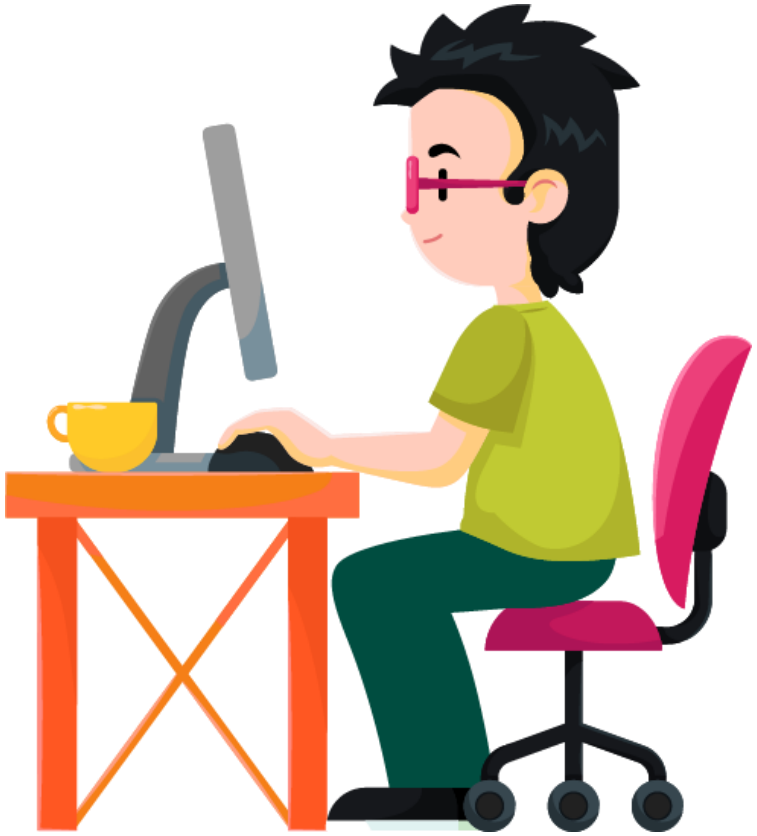
```
1   let myfunc = function(...args) {
2       console.log(args);
3   }
4
5   myfunc(3); // [3]
6   myfunc(4, 5, 6); // [4, 5, 6]
```

```
1  function demo(...args){
2
3      console.log(args);
4
5      console.log(args[0]);
6      console.log(args[1]);
7  }
8
9  demo(10,20);
```

```
[ 10, 20 ]
10
20
```

```
function demo (...arg){
    for(var i=0; i<arg.length; i++){
        console.log(arg[i]);
    }
}
demo(10,20,30,40,50);
```

Get Exclusive Video Tutorials



www.aptutorials.com
https://www.youtube.com/user/Akashtips

Get More Details

www.akashsir.com

If You Liked It !

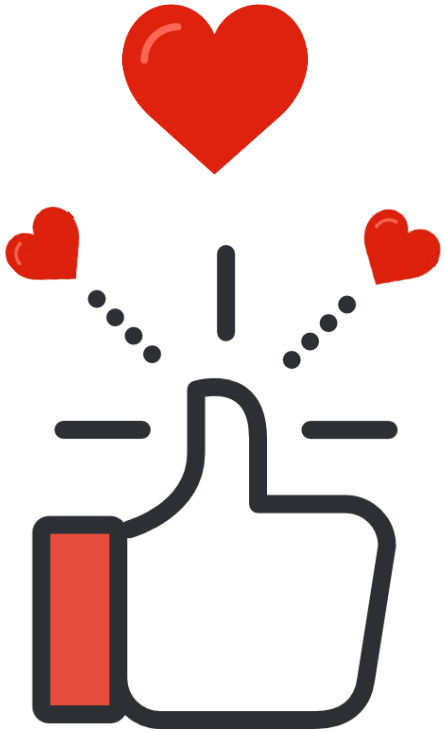
Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad



Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

Social Info

Akash.padhiyar

Akashpadhiyar

Akash_padhiyar

+91 99786-21654

#Akashpadhiyar
#aptutorials