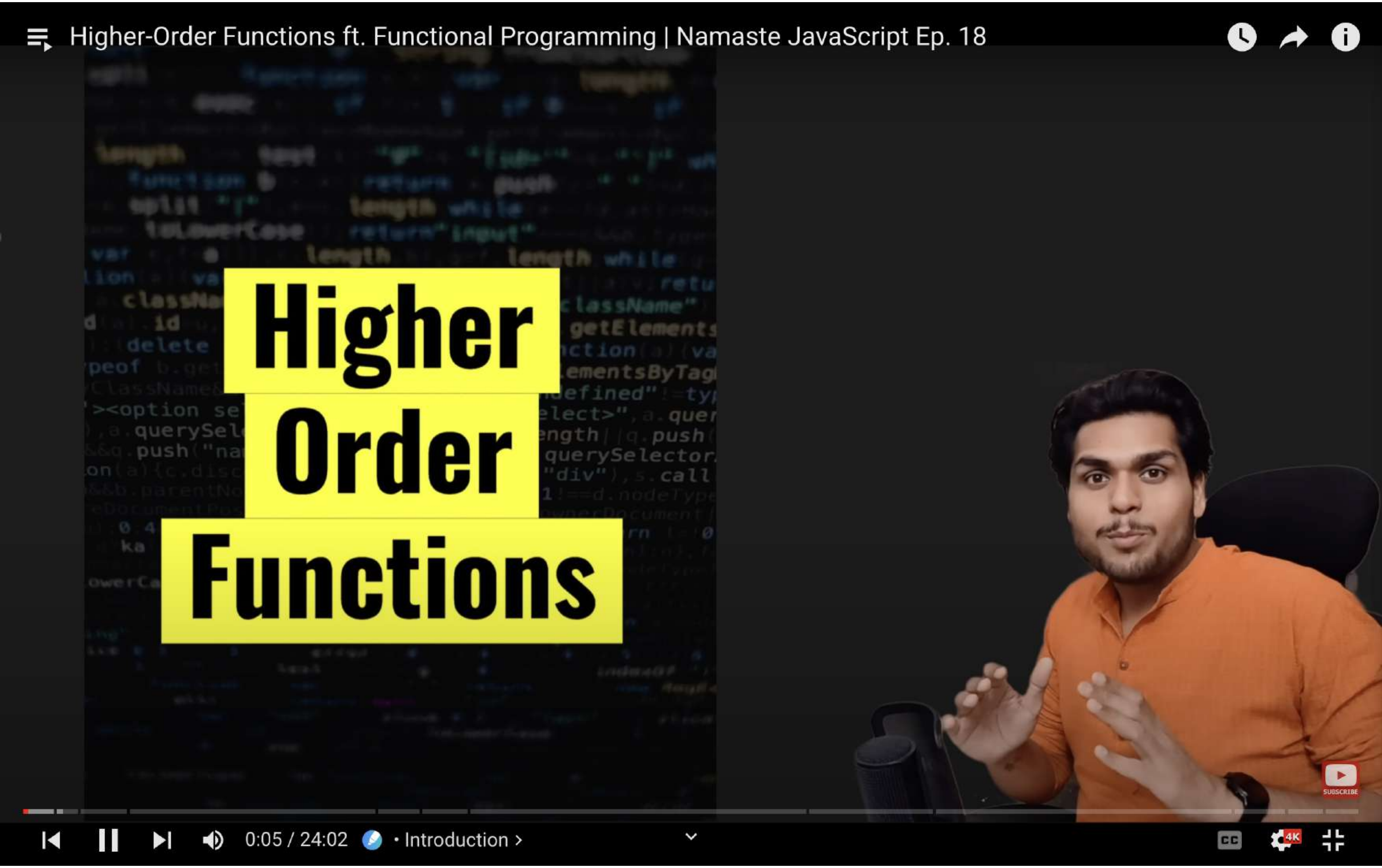
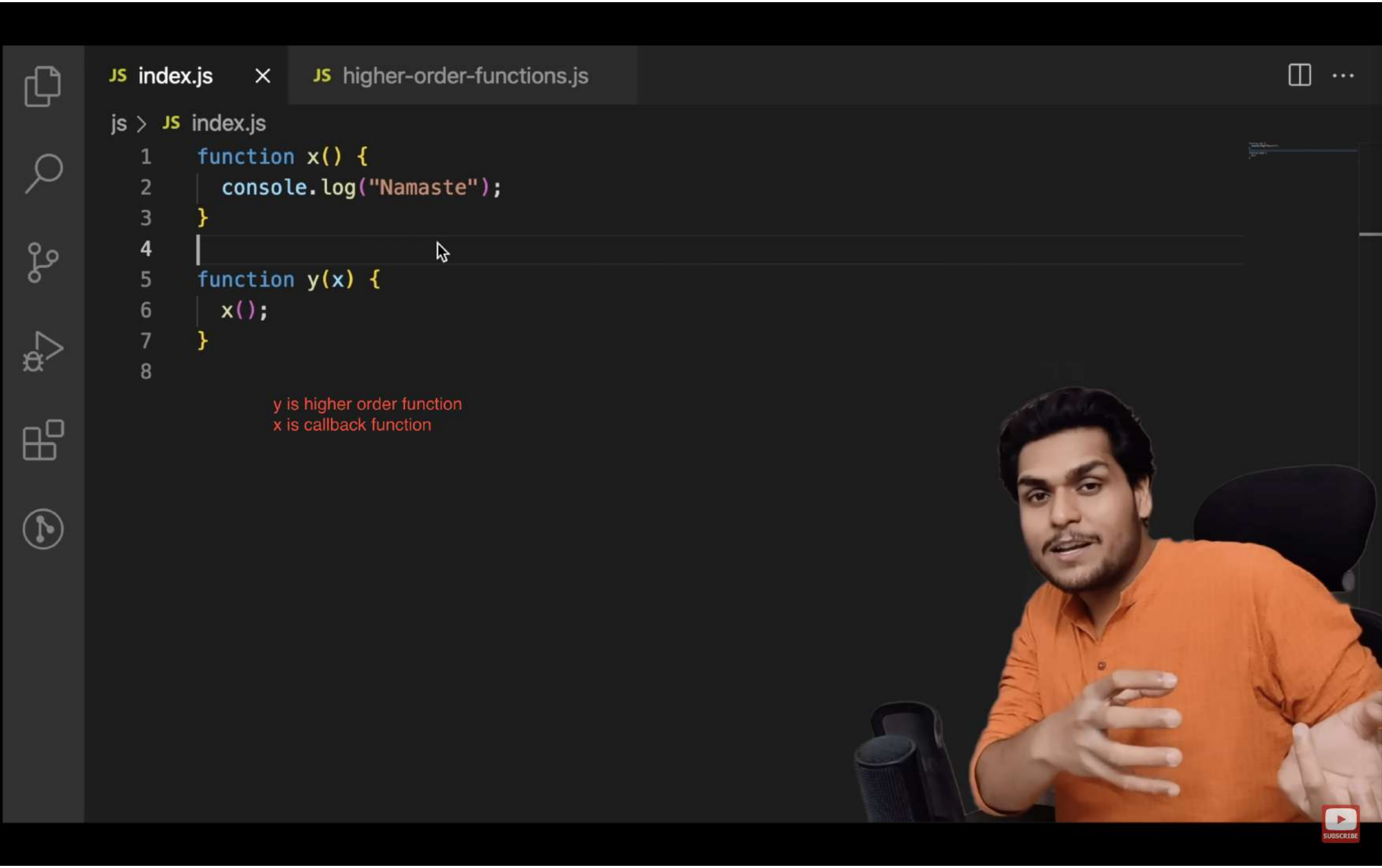
Higher-Order Functions ft. Functional Programming | Namaste JavaScript Ep. 18

Introduction to

Functional Programming


Higher-Order Functions ft. Functional Programming | Namaste JavaScript Ep. 18

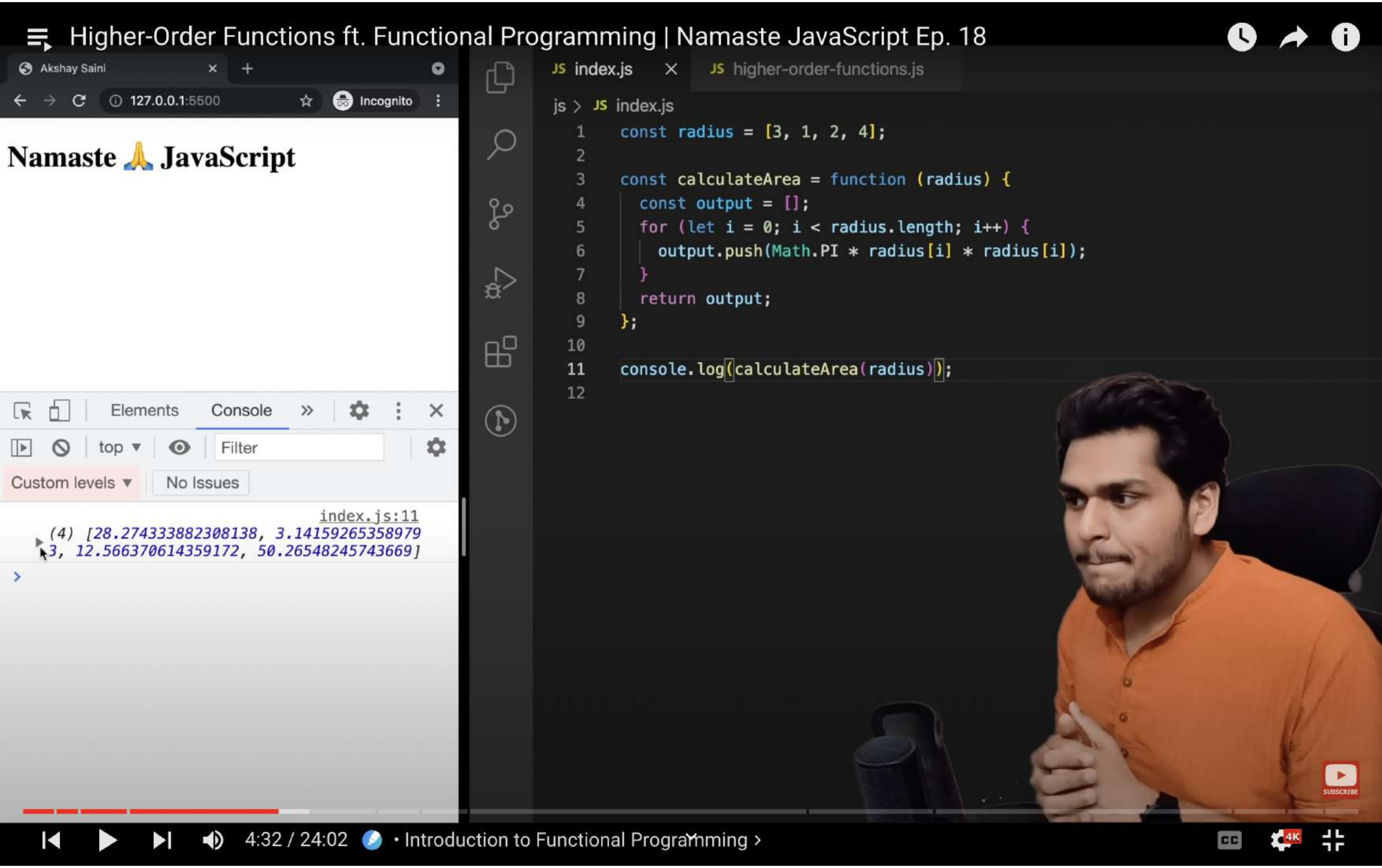


index.js
higher-order-functions.js

js > index.js

```
function x() {
  console.log("Namaste");
}

function y(x) {
  x();
}
```

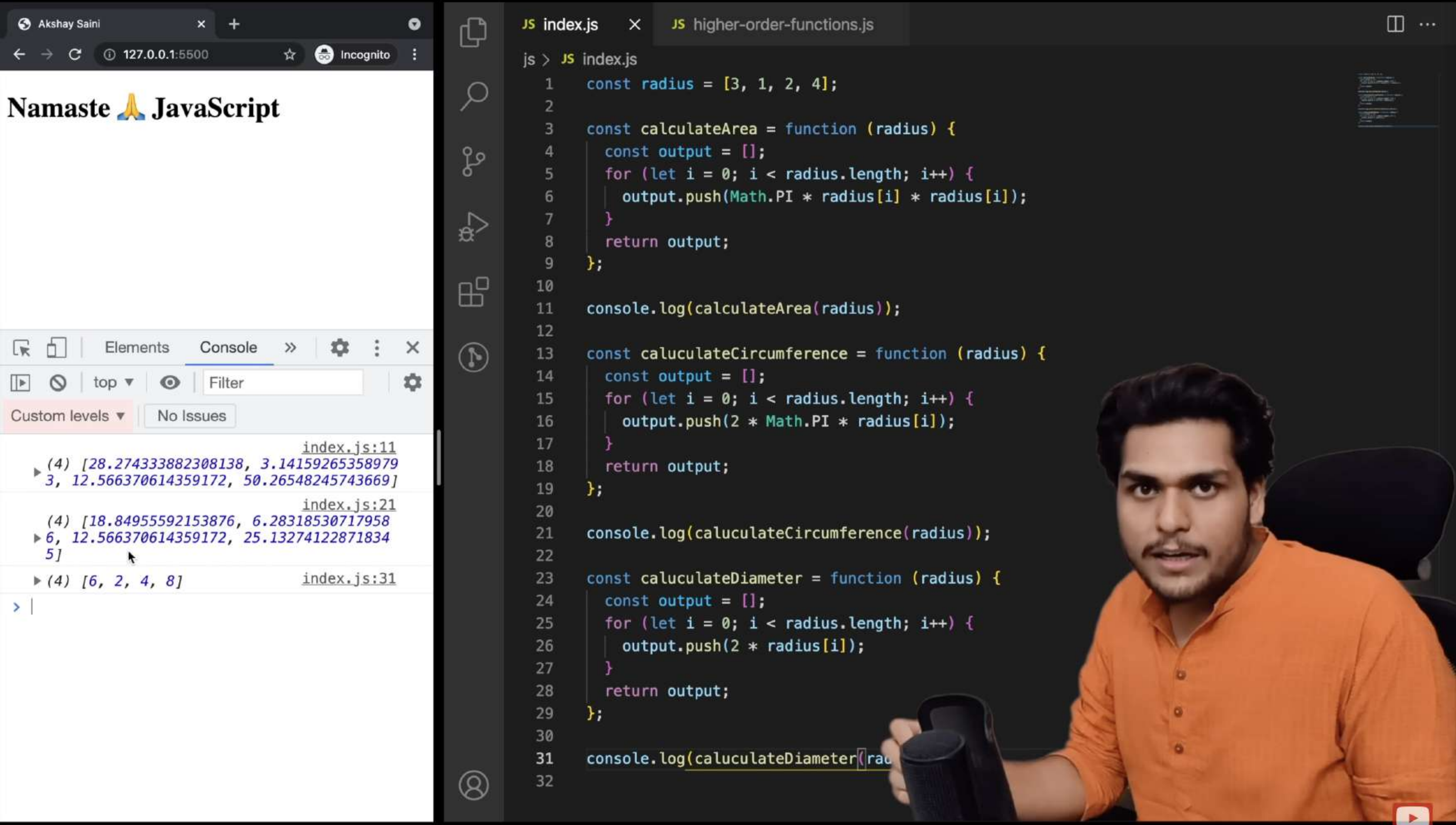
y is higher order function
x is callback function



Higher-Order Functions ft. Functional Programming | Namaste JavaScript Ep. 18

```javascript
const radius = [3, 1, 2, 4];

const calculateArea = function (radius) {
  const output = [];
  for (let i = 0; i < radius.length; i++) {
    output.push(Math.PI * radius[i] * radius[i]);
  }
  return output;
};

console.log(calculateArea(radius));
```

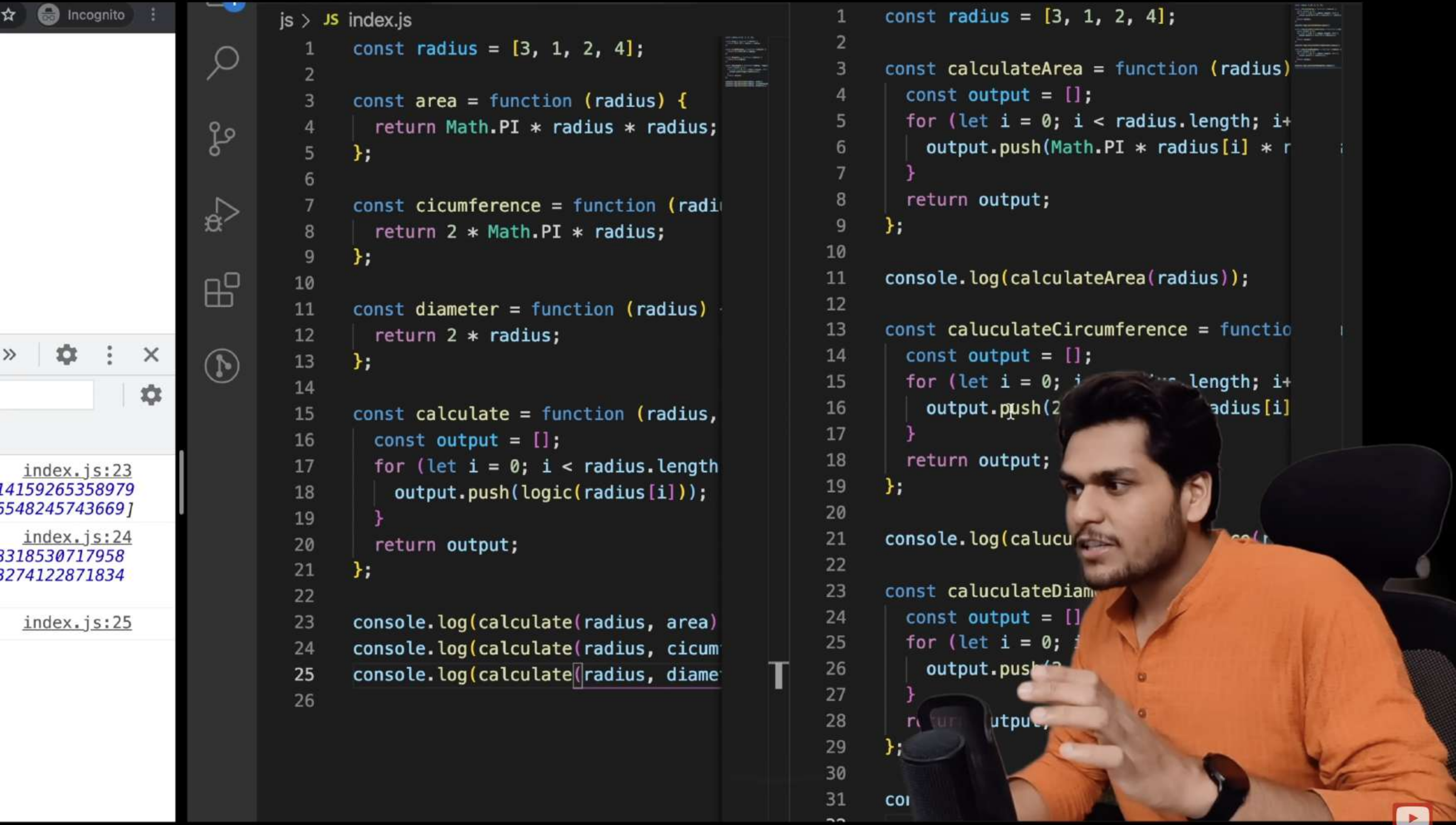
```
index.js:11
(4) [28.274333882308138, 3.141592653589793, 12.566370614359172, 50.26548245743669]
```



Namaste 🙏 JavaScript

```
index.js:11
▶ (4) [28.274333882308138, 3.141592653589793, 12.566370614359172, 50.26548245743669]
index.js:21
▶ (4) [18.84955592153876, 6.283185307179586, 12.566370614359172, 25.132741228718345]
▶ (4) [6, 2, 4, 8]                                   index.js:31
>
```

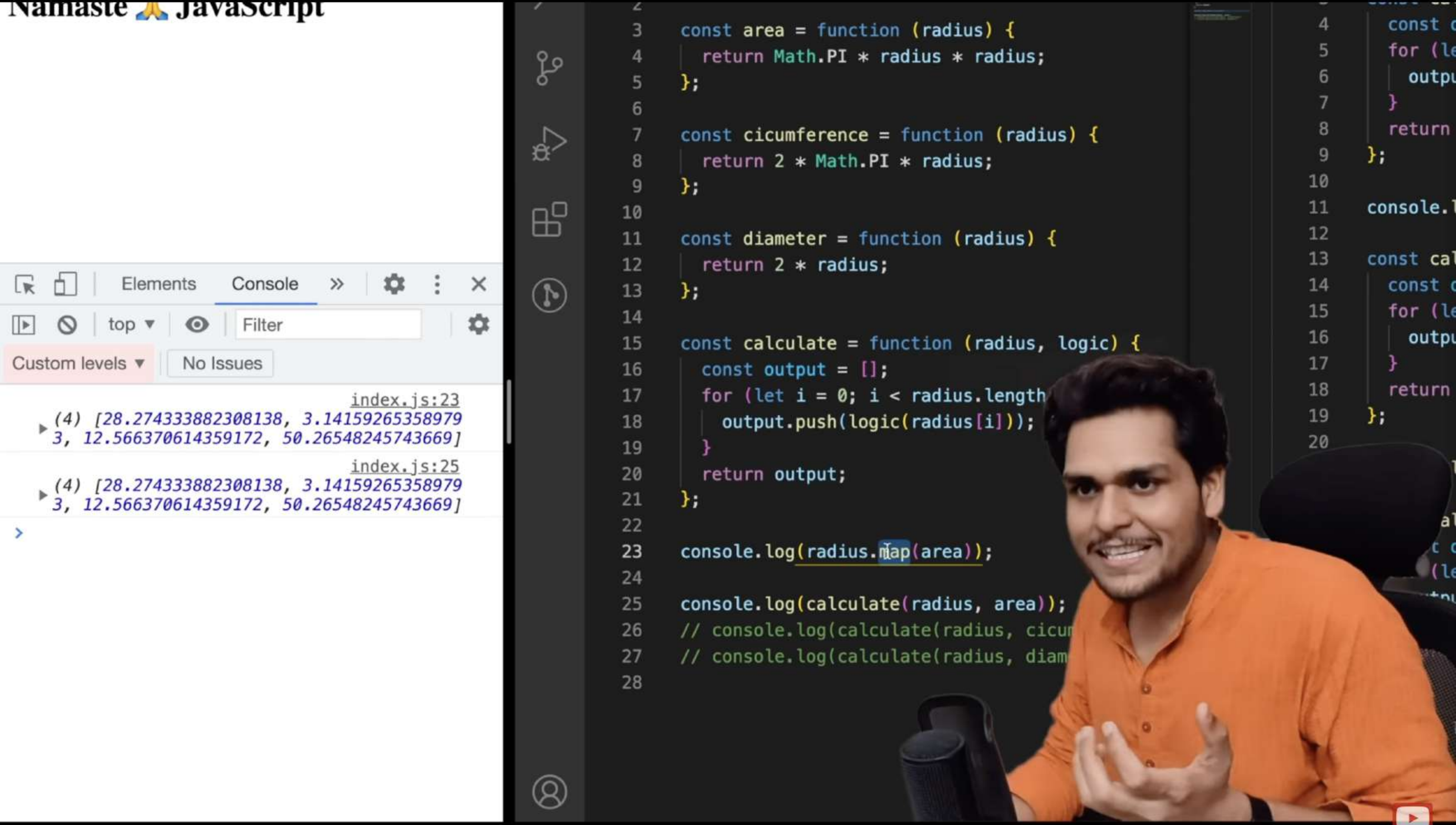
```js
const radius = [3, 1, 2, 4];

const calculateArea = function (radius) {
  const output = [];
  for (let i = 0; i < radius.length; i++) {
    output.push(Math.PI * radius[i] * radius[i]);
  }
  return output;
};

console.log(calculateArea(radius));

const caluculateCircumference = function (radius) {
  const output = [];
  for (let i = 0; i < radius.length; i++) {
    output.push(2 * Math.PI * radius[i]);
  }
  return output;
};

console.log(caluculateCircumference(radius));

const caluculateDiameter = function (radius) {
  const output = [];
  for (let i = 0; i < radius.length; i++) {
    output.push(2 * radius[i]);
  }
  return output;
};

console.log(caluculateDiameter(rad
```



```
index.js:23
14159265358979
5548245743669]
index.js:24
3318530717958
3274122871834
index.js:25
```

```
const radius = [3, 1, 2, 4];

const area = function (radius) {
  return Math.PI * radius * radius;
};

const cicumference = function (radi
  return 2 * Math.PI * radius;
};

const diameter = function (radius) 
  return 2 * radius;
};

const calculate = function (radius,
  const output = [];
  for (let i = 0; i < radius.length
    output.push(logic(radius[i]));
  }
  return output;
};

console.log(calculate(radius, area)
console.log(calculate(radius, cicum
console.log(calculate(radius, diame
```

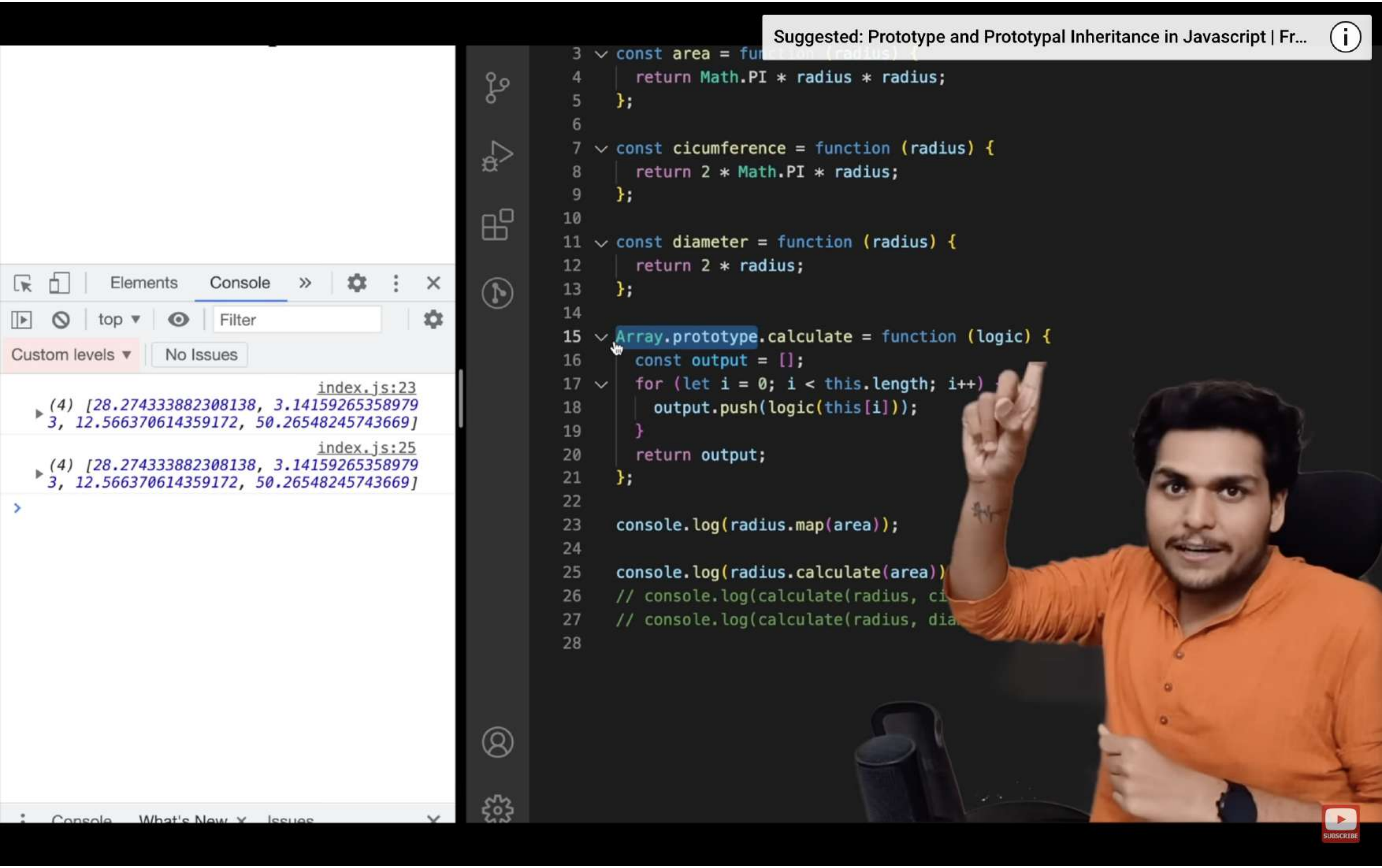
```
const radius = [3, 1, 2, 4];

const calculateArea = function (radius)
  const output = [];
  for (let i = 0; i < radius.length; i+
    output.push(Math.PI * radius[i] * r
  }
  return output;
};

console.log(calculateArea(radius));

const caluculateCircumference = functio
  const output = [];
  for (let i = 0; i          .length; i+
    output.push(2             adius[i]
  }
  return output;
};

console.log(calucu

const caluculateDiam
  const output = []
  for (let i = 0; i
    output.pus
  }
  r           utpu
};

co
```



Namaste 🙏 JavaScript

```
index.js:23
▸ (4) [28.274333882308138, 3.141592653589793, 12.566370614359172, 50.26548245743669]
index.js:25
▸ (4) [28.274333882308138, 3.141592653589793, 12.566370614359172, 50.26548245743669]
>
```

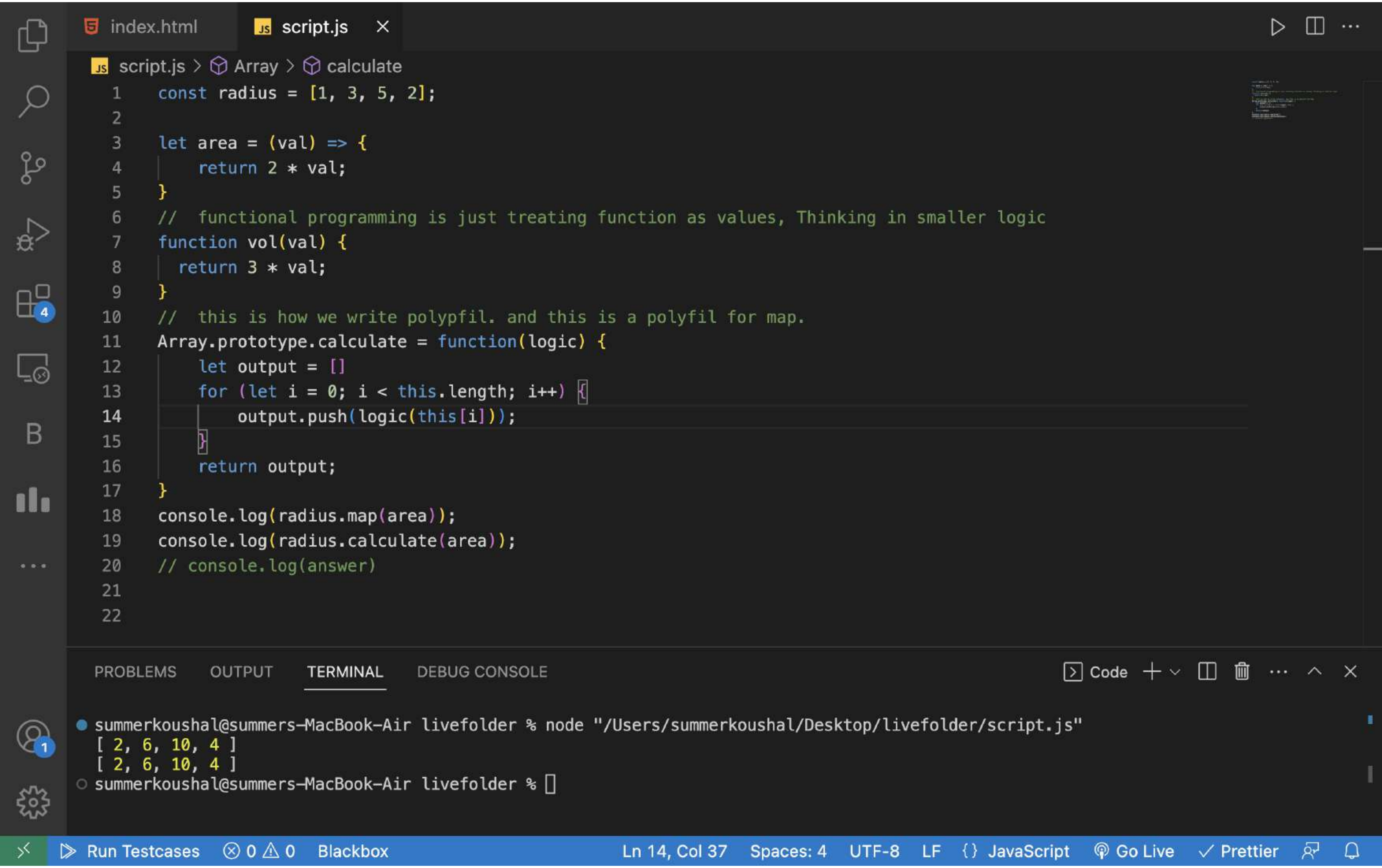
```javascript
const area = function (radius) {
  return Math.PI * radius * radius;
};

const cicumference = function (radius) {
  return 2 * Math.PI * radius;
};

const diameter = function (radius) {
  return 2 * radius;
};

const calculate = function (radius, logic) {
  const output = [];
  for (let i = 0; i < radius.length
    output.push(logic(radius[i]));
  }
  return output;
};

console.log(radius.map(area));

console.log(calculate(radius, area));
// console.log(calculate(radius, cicum
// console.log(calculate(radius, diam
```



Suggested: Prototype and Prototypal Inheritance in Javascript | Fr...

```
index.js:23
(4) [28.274333882308138, 3.141592653589793, 12.566370614359172, 50.26548245743669]
index.js:25
(4) [28.274333882308138, 3.141592653589793, 12.566370614359172, 50.26548245743669]
```

```javascript
const area = fu
  return Math.PI * radius * radius;
};

const cicumference = function (radius) {
  return 2 * Math.PI * radius;
};

const diameter = function (radius) {
  return 2 * radius;
};

Array.prototype.calculate = function (logic) {
  const output = [];
  for (let i = 0; i < this.length; i++)
    output.push(logic(this[i]));
  }
  return output;
};

console.log(radius.map(area));

console.log(radius.calculate(area))
// console.log(calculate(radius, ci
// console.log(calculate(radius, dia
```



index.html script.js

script.js > Array > calculate

```javascript
const radius = [1, 3, 5, 2];

let area = (val) => {
    return 2 * val;
}
//  functional programming is just treating function as values, Thinking in smaller logic
function vol(val) {
  return 3 * val;
}
//  this is how we write polypfil. and this is a polyfil for map.
Array.prototype.calculate = function(logic) {
    let output = []
    for (let i = 0; i < this.length; i++) {
        output.push(logic(this[i]));
    }
    return output;
}
console.log(radius.map(area));
console.log(radius.calculate(area));
// console.log(answer)


```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
summerkoushal@summers-MacBook-Air livefolder % node "/Users/summerkoushal/Desktop/livefolder/script.js"
[ 2, 6, 10, 4 ]
[ 2, 6, 10, 4 ]
summerkoushal@summers-MacBook-Air livefolder % 
```

not possible without

Higher Order functions
(functional programming)

One of the most amazing part of javascript.

Higher Order functions.
A function taking another function as an argument or returns a function from it.

Example:
```
        function x(){
            ...
        }
        function y(x){
            x();
        }
```
Y → Higher Order function
x → callback function

functional programming says:
- make logic according to functions.
- ✓ Reusability
- ✓ Modularity

higher Order function + callback function
↳ functional programming

Our calculate func. is very similar to map function.