# ES6+ Exercises

# section 1: let, const, and var

## example 01

Topic: Block scope & functional scope

What will be the output and why?

COPY

```
if (true) {
  let a = 2
  // var a = 2;
}
console.log(a)
```

What will be the output and why?

COPY

```
let a = 42
{
  let a = 100
}
console.log(a)
```

What will be the output and why?

COPY

```
let a = 100
function App() {
  console.log('1', a)
  let a = 42
  console.log('2', a)
  {
    let a = 100
  }
  console.log('3', a)
}
```

What will be the output and why?

COPY

```
let a = 100
function App() {
  a = 42
  console.log('1', a)
}
```

## example 02

Topic: TDZ. Temporal Dead Zone What will be the output of this? And why?

COPY

```
function something() {
  console.log(a)
  let a = 2
}

something()
```

## example 03

Topic: Hoisting.

Output and why?

COPY

```
function something() {
  // a = undefined
  console.log(a)
  var a = 2
}

something()
```

## example 04

Topic: const VS var

Output and why?

```
const tanay = 2
tanay = 3
```

COPY

## example 05

Topic: const with objects

Will we get an error?

```
const obj = { a: 1, b: 2 }
obj.a = 22
```

COPY

Will we get an error?

```
const obj = { a: 1, b: 2 }
const obj2 = { a: 3, b: 4 }
obj = obj2
```

COPY

## example 06

Topic: const with arrays

Will we get an error?

What will it return?

```
const array = [1, 2, 3, 4]
array.push(55)
```

COPY

Will we get an error?

```
const array = [1, 2, 3, 4]
array = array.push(55)
```

COPY

# section 2: arrow functions

## example 07

Topic: Short arrow function/Lamda function/Fat arrow function

What will be the output?

Can you convert below arrow function into function declaration (normal function) format?

COPY

```
// with no parameter
const getNum = () => 2

// ES5 function
function getNum() {
  return 2
}

// with one parameter
// Function
const isOne = (num) => num == 1 // HW: === vs == in JavaScript, write a blog.

// Calling the function
console.log(isOne(12))

// Vinay

function isOne(num) {
  return num === 1 ? true : false
}

// Hiren
function isOne(num) {
  console.log(num)
}
```

## example 08

## Topic: Arrow function with more than 1 parameter

COPY

```
const isBigger = (a, b) => a > b
console.log('Is 2 bigger than 3?', isBigger(2, 3))

// How would you write this with the least amount of characters
function Add22andReturn(num) {
  let sum = 0
  sum = num + 22
  return sum
}

// Version 1
const Add22andReturn = (num) => {
  let sum = 0
  sum = num + 22
  return sum
}

// Version 2
const Add22andReturn = (num) => {
  let sum = 0
  sum = num + 22
  return sum
}

// Version 3
const Add22andReturn = (num) => {
  return num + 22
}

// Version 4
const Add22andReturn = (num) =>
```

```
  num +
  (22)[
    // Doing it on an array
    (1, 2, 3, 4, 5)
  ].map((num) => num + 22)
```

## example 09

Topic: Arrow function when there are more than 1 statement

```
const printSomethingAndReturn = (a) => {
  console.log('squaring a.....')
  return a * a
}
console.log(printSomethingAndReturn(2))
```

COPY

## example 10

Topic: Short arrow funtion while returning an object / object literal syntax

```
const giveMeAnObject = (a) => ({ value: a })

console.log(giveMeAnObject(5))
```

COPY

# section 3: default parameters

## example 11

Topic: default parameters

What will be the output?

```
const defaultExample = (a, b) => a + b
console.log(defaultExample(2))
```

COPY

Will this work?

```
const defaultExample = (a, b) => {
  if (b === undefined) {
    b = 5
  }
  return a + b
}

// Instead
const defaultExample2 = (a, b = 5) => a + b

console.log(defaultExample(2, 4))
console.log(defaultExample2(3))
```

COPY

What will be the output?

Will we get an error?

```
const defaultExample3 = (a, b=5, c) => a + b + c

console.log(defaultExample3(1, 2, 3));
console.log(defaultExample3(1, , 3));
```

COPY

# section 4: rest and spread

## example 12

Topic: rest

```
const restExample = (a, ...rest) => {
  console.log(a, rest)
}

console.log(restExample(2, 3, 4, 5, 6, 7, 8, 9, 10))
```

COPY

## example 13

Topic: spread with object

```
const spreadExample = ({ a, b }) => {
  console.log(a, b)
}

const obj = { a: 1, b: 2 }
console.log(spreadExample(obj))
```

COPY

## example 14

Topic: spread with array

```
const array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
const spreadExample2 = ([first, second, ...rest]) => {
  console.log(first, second, rest)
}

console.log(spreadExample2(array))

// rest with spread
console.log(spreadExample2([...array, 11, 12, 13]))
```

COPY

## example 15

Topic: spread ( can rename parameters )

Will we get an error?

```
const spreadExample = ({ a: aaloo, b: bhaaloo }) => {
  // destructuring
  // console.log(a,b)
  console.log(aaloo, bhaaloo)
}

const obj = { a: 1, b: 2 }
spreadExample(obj)
```

COPY

What will be the output?

```
const spreadExample = ({ b: bhaaloo, a: aaloo }) => {
  console.log(aaloo, bhaaloo)
}

const obj = { a: 1, b: 2 }
spreadExample(obj)
```

COPY

# section 5: Dynamic fields/ Object literal/ Template literal

## example 16

Topic: Dynamic fields in an object/computed property syntax

COPY

```
let key = 'ram'
const obj = { [key]: 122 }
console.log(obj)
```

## example 17

Topic: Object literals/Shorthand property value

COPY

```
let aaloo = 1
let bhaaloo = 2

const obj1 = { aaloo: aaloo, bhaaloo: bhaaloo }
console.log(obj1)
const obj2 = { aaloo, bhaaloo }
console.log(obj2) // { aaloo: 1, bhaaloo: 2}
```

## example 18

Topic: Template Literal

COPY

```
let name = 'Tanay'
let line = 'Helloo ' + name + ' !'
console.log(line)

const helloTemp = `Hello ${name} !`
console.log(helloTemp)

const giveMeFive = () => 5
const line2 = `Hey, my roll is ${giveMeFive()}`
console.log(line2)

// Question: variable 5 // "odd" ->> write this function, using arrow
// Question: use this function in a sentence using template literal

// Shivam's answer
const oddOrEven = (num) => (num % 2 === 0 ? 'even' : 'odd')
const res = `The number 5 is ${oddOrEven(5)} ${name}`
```

# section 6: Module system

## example 19

Topic: import/export

COPY

```
//  →  /utils.js
export const add = (a, b) => a + b
export const square = (a) => a * a
```

COPY

```
// → /app.js
import { add, square } from './utils'

console.log(add(1, 2))
console.log(square(2))
```

## example 20

Topic: default export

COPY

```
// → /utils.js

const square = (a) => a * a
export default square
```

COPY

```
// → /app.js
import square from './utils'

console.log(square(2))
```

## example 21

Topic: import/export with rename

```
// → /utils.js

export { add as sum }
```

COPY



```
// → /app.js

import { sum as xyz } from './utils'

console.log(xyz(2, 5))
```

COPY

Previous
‹ ES6+ Session

Next
ES6+ Post-Reads ›