NodeJS

LABA

FUNCTIONS

FUNCTIONS

```
function doSomething(foo) {}

const doSomething = function(foo) {}

const doSomething = foo => {}
```

Solvd
NodeJS LABA

FUNCTIONS

```
const doSomething = () => {}

const doSomethingElse = foo => {}

const doSomethingElseAgain = (foo, bar) => {}

const doSomething = (foo = 1, bar = 'hey') => {}
```

FUNCTIONS

```
doSomething(3)
doSomething()

const doSomething = (foo = 1, bar = 'hey') => {}
const args = [2, 'ho!']
const doSomething(...args)
```

Solvd
NodeJS LABA

FUNCTIONS

```
const doSomething = ({ foo = 1, bar = 'hey' }) => {
  console.log(foo)
  console.log(bar)
}

const args = { foo: 2, bar: 'ho!' }
doSomething(args);
```

Solvd
NodeJS LABA

FUNCTIONS

```
const doSomething = (foo = 1, bar = 'hey') => {}

console.log(doSomething())

const doSomething = () => {
  return 'test'
}

const result = doSomething() // result === 'test'
```

FUNCTIONS

```
const doSomething = () => {
  return ['Roger', 6]
}

const [name, age] = doSomething()
console.log(name, age) // Roger 6

const doSomething = () => {
  return {name: 'Roger', age: 6}
}

const { name, age } = doSomething()
console.log(name, age) // Roger 6
```

Solvd
NodeJS LABA

FUNCTIONS

```
const doSomething = () => {
  const doSomethingElse = () => {}
  doSomethingElse()
  return 'test'
}

doSomething()
```

Solvd
NodeJS LABA

FUNCTIONS

```
const car = {
  brand: 'Ford',
  model: 'Fiesta',
  start: function() {
    console.log('Started')
  }
}

car.start()
```

FUNCTIONS

```
const car = {
  brand: 'Ford',
  model: 'Fiesta',
  start: function() {
    console.log(`Started ${this.brand} ${this.model}`)
  },
  stop: () => {
    console.log(`Stopped ${this.brand} ${this.model}`)
  }
}

car.start() // Started Ford Fiesta
car.stop() // Stopped undefined undefined
```

FUNCTIONS

```
const test = {
  fn: function() {
    console.log(this)
  },
  arrFn: () => {
    console.log(this)
  }
}

test.fn()
test.arrFn()
```

FUNCTIONS

```
;(function () {
  console.log('executed')
})()
```

Solvd
NodeJS LABA

FUNCTIONS

```
const something = (function() {
  return 'IIFE'
})()

console.log(something)
```

Solvd
NodeJS LABA

FUNCTIONS

```
doSomething() // did something
function doSomething() {
  console.log('did something')
}

doSomething() // TypeError
var doSomething = function() {
  console.log('did something')
}
```

FUNCTIONS

```
const myFunction = function() {}

const myFunction = () => {}

const myFunction = (a, b) => a + b

const myFunction = a => a / 2
console.log(myFunction(8)) // 4
```

FUNCTIONS

```
const myFunction = () => ({ value: 'test' })

const obj = myFunction()
console.log(obj.value) // test
```

Solvd
NodeJS LABA

FUNCTIONS

```
const bark = dog => {
  const say = `${dog} barked!`
  ;(() => console.log(say))()
}
bark(`Roger`) // Roger barked!
```

Solvd
NodeJS LABA

FUNCTIONS

```
const prepareBark = dog => {
  const say = `${dog} barked!`
  return () => console.log(say)
}

const bark = prepareBark(`Roger`)
bark()
```

Solvd NodeJS LABA

FUNCTIONS

```
const prepareBark = dog => {
  const say = `${dog} barked!`
  return () => {
    console.log(say)
  }
}

const rogerBark = prepareBark(`Roger`)
const sydBark = prepareBark(`Syd`)
rogerBark()
sydBark()
```

Solvd
NodeJS LABA

FUNCTIONS

```
const circleArea(radius) {
  return radius * radius * Math.PI
}

const counter = (function() {
  let initValue = 0
  return function() {
    initValue++;
    return initValue
  }
})()
```

Solvd
NodeJS LABA

FUNCTIONS

```
let femaleCounter = 0;
let maleCounter = 0;
function isMale(user) {
  if(user.sex = 'man') {
    maleCounter++;
    return true
  }
  return false
}
```

FUNCTIONS

```
for (let i = 0; i < 1000; i++) {
  console.log(fun(10));
}

let result = fun(10)
for (let i = 0; i < 1000; i++) {
  console.log(result);
}

const incrementNumbers = function(numbers) {}
```

Solvd
NodeJS LABA

FUNCTIONS

```
let list = [1, 2, 3, 4, 5]
assert.equals(incrementNumbers(list), [2, 3, 4, 5, 6])
```

Solvd
NodeJS LABA

FUNCTIONS

```
const arr1 = [1, 2, 3];
const arr2 = [];
for (let i = 0; i < arr1.length; i++) {
  arr2.push(arr[i] * 2);
}

const arr1 = [1, 2, 3];
const arr2 = arr1.map(function(item) {
  return item * 2;
})
console.log(arr2);
```

FUNCTIONS

```
function computed(str) {
  console.log('2000s have passed')

  return 'a result'
}

function cached(fn) {
  const cache = Object.create(null);
  return function cachedFn(str) {
    if (!cache[str]) {
      let result = fn(str);
      cache[str] = result;
    }

    return cache[str]
  }
}
```

FUNCTIONS

```
let fooFirstExecutedDate = null;
function foo() {
  if (fooFirstExecutedDate != null) {
    return fooFirstExecutedDate;
  } else {
    fooFirstExecutedDate = new Date();
    return fooFirstExecutedDate;
  }
}
```

Solvd
NodeJS LABA

FUNCTIONS

```
function add(a, b, c) {
  return a + b + c;
}

add(1, 2, 3) // 6
add(1, 2) // NaN
add(1, 2, 3, 4) // 6
```

Solvd
NodeJS LABA

FUNCTIONS

```
function curry(fn) {
  if (fn.length <= 1) return fn;

  const generator = (...args) => {
    if (fn.length === args.length) {
      return fn(...args)
    } else {
      return (...args2) => {
        return generator(...args, ...args2)
      }
    }
  }
  return generator
}
```

Solvd
NodeJS LABA

FUNCTIONS

```
let toUpperCase = function(x) { return x.toUpperCase(); }

let hello = function(x) { return 'HELLO, ' + x; }

let greet = function(x) {
  return hello(toUpperCase(x));
}

let compose = function(f, g) {
  return function(x) {
    return f(g(x));
  }
}

let greet = compose(hello, toUpperCase);
greet('kevin')
```

FUNCTIONS

```
function compose() {
  var args = arguments;
  var start = args.length - 1;
  return function() {
    var i = start;
    var result = args[start].apply(this, arguments);
    while (i--) result = args[i].call(this, result);
    return result;
  }
}
```

Solvd
NodeJS LABA