完整面试题地址：https://interview.poetries.top

作者：程序员poetry

扫码关注作者公众号：「前端进价之旅」 每天分享技术干货

> `JavaScript` 进阶问题列表

进阶问题

# 1. 输出是什么?

```js
function sayHi() {
  console.log(name)
  console.log(age)
  var name = 'Lydia'
  let age = 21
}

sayHi()
```

- A: `Lydia` 和 `undefined`
- B: `Lydia` 和 `ReferenceError`
- C: `ReferenceError` 和 `21`
- D: `undefined` 和 `ReferenceError`

▶ **答案**

# 2. 输出是什么?

```js
for (var i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), 1)
}
```

```js
for (let i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), 1)
}
```

- A: `0 1 2` 和 `0 1 2`
- B: `0 1 2` 和 `3 3 3`
- C: `3 3 3` 和 `0 1 2`

▶ **答案**

# 3. 输出是什么?

```js
const shape = {
  radius: 10,
  diameter() {
    return this.radius * 2
  },
  perimeter: () => 2 * Math.PI * this.radius
}

shape.diameter()
shape.perimeter()
```

- A: `20` and `62.83185307179586`
- B: `20` and `NaN`
- C: `20` and `63`
- D: `NaN` and `63`

▶ **答案**

# 4. 输出是什么?

```js
;+true
!'Lydia'
```

- A: `1` and `false`
- B: `false` and `NaN`
- C: `false` and `false`

▶ **答案**

# 5. 哪一个是无效的?

```js
const bird = {
  size: 'small'
}

const mouse = {
  name: 'Mickey',
  small: true
}
```

- A: `mouse.bird.size`
- B: `mouse[bird.size]`
- C: `mouse[bird["size"]]`
- D: All of them are valid

▶ **答案**

# 6. 输出是什么?

```js
let c = { greeting: 'Hey!' }
let d
```

```
d = c
c.greeting = 'Hello'
console.log(d.greeting)
```

- A: `Hello`
- B: `undefined`
- C: `ReferenceError`
- D: `TypeError`

▶ **答案**

7. 输出是什么?

```js
 let a = 3
let b = new Number(3)
let c = 3

console.log(a == b)
console.log(a === b)
console.log(b === c)
```

- A: `true` `false` `true`
- B: `false` `false` `true`
- C: `true` `false` `false`
- D: `false` `true` `true`

▶ **答案**

8. 输出是什么?

```js
class Chameleon {
  static colorChange(newColor) {
    this.newColor = newColor
    return this.newColor
  }

  constructor({ newColor = 'green' } = {}) {
    this.newColor = newColor
  }
}

const freddie = new Chameleon({ newColor: 'purple' })
freddie.colorChange('orange')
```

- A: `orange`
- B: `purple`
- C: `green`
- D: `TypeError`

▶ **答案**

# 9. 输出是什么?

```js
let greeting
greetign = {} // Typo!
console.log(greetign)
```

- A: `{}`
- B: `ReferenceError: greetign is not defined`
- C: `undefined`

▶ **答案**

# 10. 当我们这么做时，会发生什么？

```js
function bark() {
  console.log('Woof!')
}

bark.animal = 'dog'
```

- A: 正常运行!
- B: `SyntaxError` . 你不能通过这种方式给函数增加属性。
- C: `undefined`
- D: `ReferenceError`

▶ **答案**

# 11. 输出是什么？

```js
function Person(firstName, lastName) {
  this.firstName = firstName
  this.lastName = lastName
}

const member = new Person('Lydia', 'Hallie')
Person.getFullName = function () {
  return `${this.firstName} ${this.lastName}`
}

console.log(member.getFullName())
```

- A: `TypeError`
- B: `SyntaxError`
- C: `Lydia Hallie`

- D: `undefined` `undefined`

▶ **答案**

# 12. 输出是什么?

```js
function Person(firstName, lastName) {
  this.firstName = firstName
  this.lastName = lastName
}

const lydia = new Person('Lydia', 'Hallie')
const sarah = Person('Sarah', 'Smith')

console.log(lydia)
console.log(sarah)
```

- A: `Person {firstName: "Lydia", lastName: "Hallie"}` and `undefined`
- B: `Person {firstName: "Lydia", lastName: "Hallie"}` and `Person {firstName: "Sarah", lastName: "Smith"}`
- C: `Person {firstName: "Lydia", lastName: "Hallie"}` and `{}`
- D: `Person {firstName: "Lydia", lastName: "Hallie"}` and `ReferenceError`

▶ **答案**

# 13. 事件传播的三个阶段是什么?

- A: Target > Capturing > Bubbling
- B: Bubbling > Target > Capturing
- C: Target > Bubbling > Capturing
- D: Capturing > Target > Bubbling

▶ **答案**

# 14. 所有对象都有原型。

- A: true
- B: false

▶ **答案**

# 15. 输出是什么?

```js
function sum(a, b) {
  return a + b
}

sum(1, '2')
```

- A: `NaN`
- B: `TypeError`
- C: `"12"`
- D: `3`

▶ **答案**

# 16. 输出是什么?

```js
let number = 0
console.log(number++)
console.log(++number)
console.log(number)
```

- A: `1` `1` `2`

- B: `1` `2` `2`
- C: `0` `2` `2`
- D: `0` `1` `2`

▶ **答案**

# 17. 输出是什么?

```js
function getPersonInfo(one, two, three) {
  console.log(one)
  console.log(two)
  console.log(three)
}

const person = 'Lydia'
const age = 21

getPersonInfo`${person} is ${age} years old`
```

- A: `"Lydia"` `21` `["", " is ", " years old"]`
- B: `["", " is ", " years old"]` `"Lydia"` `21`
- C: `"Lydia"` `["", " is ", " years old"]` `21`

▶ **答案**

# 18. 输出是什么?

```js
function checkAge(data) {
  if (data === { age: 18 }) {
    console.log('You are an adult!')
  } else if (data == { age: 18 }) {
    console.log('You are still an adult.')
  } else {
    console.log(`Hmm.. You don't have an age I guess`)
  }
}

checkAge({ age: 18 })
```

- A: `You are an adult!`
- B: `You are still an adult.`
- C: `Hmm.. You don't have an age I guess`

▶ **答案**

# 19. 输出是什么?

```js
function getAge(...args) {
  console.log(typeof args)
}

getAge(21)
```

- A: `"number"`
- B: `"array"`
- C: `"object"`
- D: `"NaN"`

▶ **答案**

# 20. 输出是什么?

```js
function getAge() {
  'use strict'
  age = 21
  console.log(age)
}
```

```
getAge()
```

- A: `21`
- B: `undefined`
- C: `ReferenceError`
- D: `TypeError`

▶ **答案**

# 21. 输出是什么?

```js
const sum = eval('10*10+5')
```

- A: `105`
- B: `"105"`
- C: `TypeError`
- D: `"10*10+5"`

▶ **答案**

# 22. cool_secret 可访问多长时间?

```js
sessionStorage.setItem('cool_secret', 123)
```

- A: 永远，数据不会丢失。
- B: 当用户关掉标签页时。
- C: 当用户关掉整个浏览器，而不只是关掉标签页。
- D: 当用户关闭电脑时。

▶ **答案**

# 23. 输出是什么?

```js
var num = 8
var num = 10

console.log(num)
```

- A: `8`
- B: `10`
- C: `SyntaxError`
- D: `ReferenceError`

▶ **答案**

# 24. 输出是什么?

```js
const obj = { 1: 'a', 2: 'b', 3: 'c' }
const set = new Set([1, 2, 3, 4, 5])

obj.hasOwnProperty('1')
obj.hasOwnProperty(1)
set.has('1')
set.has(1)
```

- A: `false` `true` `false` `true`
- B: `false` `true` `true` `true`
- C: `true` `true` `false` `true`
- D: `true` `true` `true` `true`

▶ **答案**

# 25. 输出是什么?

```js
const obj = { a: 'one', b: 'two', a: 'three' }
console.log(obj)
```

- A: `{ a: "one", b: "two" }`
- B: `{ b: "two", a: "three" }`
- C: `{ a: "three", b: "two" }`
- D: `SyntaxError`

▶ **答案**

# 26. JavaScript 全局执行上下文为你做了两件事：全局对象和 this 关键字。

- A: true
- B: false
- C: it depends

▶ **答案**

# 27. 输出是什么?

```js
for (let i = 1; i < 5; i++) {
  if (i === 3) continue
  console.log(i)
}
```

- A: `1` `2`

- B: `1` `2` `3`
- C: `1` `2` `4`
- D: `1` `3` `4`

▶ **答案**

# 28. 输出是什么?

```js
String.prototype.giveLydiaPizza = () => {
  return 'Just give Lydia pizza already!'
}

const name = 'Lydia'

name.giveLydiaPizza()
```

- A: `"Just give Lydia pizza already!"`
- B: `TypeError: not a function`
- C: `SyntaxError`
- D: `undefined`

▶ **答案**

# 29. 输出是什么?

```js
const a = {}
const b = { key: 'b' }
const c = { key: 'c' }

a[b] = 123
a[c] = 456

console.log(a[b])
```

- A: `123`
- B: `456`
- C: `undefined`
- D: `ReferenceError`

▶ **答案**

# 30. 输出是什么?

```js
const foo = () => console.log('First')
const bar = () => setTimeout(() => console.log('Second'))
const baz = () => console.log('Third')

bar()
foo()
baz()
```

- A: `First` `Second` `Third`
- B: `First` `Third` `Second`
- C: `Second` `First` `Third`
- D: `Second` `Third` `First`

▶ **答案**

# 31. 当点击按钮时，event.target是什么?

```html
<div onclick="console.log('first div')">
  <div onclick="console.log('second div')">
    <button onclick="console.log('button')">Click!</button>
  </div>
</div>
```

- A: Outer `div`
- B: Inner `div`
- C: `button`
- D: 一个包含所有嵌套元素的数组。

▶ **答案**

# 32. 当您单击该段落时，日志输出是什么？

```html
<div onclick="console.log('div')">
  <p onclick="console.log('p')">Click here!</p>
</div>
```

- A: `p` `div`
- B: `div` `p`
- C: `p`
- D: `div`

▶ **答案**

# 33. 输出是什么？

```js
const person = { name: 'Lydia' }

function sayHi(age) {
  console.log(`${this.name} is ${age}`)
}

sayHi.call(person, 21)
sayHi.bind(person, 21)
```

- A: `undefined is 21` `Lydia is 21`

- B: `function` `function`
- C: `Lydia is 21` `Lydia is 21`
- D: `Lydia is 21` `function`

▶ **答案**

# 34. 输出是什么?

```js
 function sayHi() {
   return (() => 0)()
}

typeof sayHi()
```

- A: `"object"`
- B: `"number"`
- C: `"function"`
- D: `"undefined"`

▶ **答案**

# 35. 下面哪些值是 falsy?

```js
 0
new Number(0)('')(' ')
new Boolean(false)
undefined
```

- A: `0` , `''` , `undefined`
- B: `0` , `new Number(0)` , `''` , `new Boolean(false)` , `undefined`
- C: `0` , `''` , `new Boolean(false)` , `undefined`

- D: All of them are falsy

▶ **答案**

# 36. 输出是什么?

```js
console.log(typeof typeof 1)
```

- A: `"number"`
- B: `"string"`
- C: `"object"`
- D: `"undefined"`

▶ **答案**

# 37. 输出是什么?

```js
const numbers = [1, 2, 3]
numbers[10] = 11
console.log(numbers)
```

- A: `[1, 2, 3, 7 x null, 11]`
- B: `[1, 2, 3, 11]`
- C: `[1, 2, 3, 7 x empty, 11]`
- D: `SyntaxError`

▶ **答案**

# 38. 输出是什么?

```js
;(() => {
  let x, y
  try {
    throw new Error()
  } catch (x) {
    ;(x = 1), (y = 2)
    console.log(x)
  }
  console.log(x)
  console.log(y)
})()
```

- A: `1` `undefined` `2`
- B: `undefined` `undefined` `undefined`
- C: `1` `1` `2`
- D: `1` `undefined` `undefined`

▶ **答案**

# 39. JavaScript 中的一切都是?

- A: 基本类型与对象
- B: 函数与对象
- C: 只有对象
- D: 数字与对象

▶ **答案**

# 40. 输出是什么?

```js
;[
  [0, 1],
  [2, 3]
].reduce(
  (acc, cur) => {
    return acc.concat(cur)
  },
  [1, 2]
)
```

- A: `[0, 1, 2, 3, 1, 2]`
- B: `[6, 1, 2]`
- C: `[1, 2, 0, 1, 2, 3]`
- D: `[1, 2, 6]`

▶ **答案**

# 41. 输出是什么?

```js
!!null
!!''
!!1
```

- A: `false` `true` `false`
- B: `false` `false` `true`
- C: `false` `true` `true`
- D: `true` `true` `false`

▶ **答案**

# 42. `setInterval` 方法的返回值是什么?

```js
setInterval(() => console.log('Hi'), 1000)
```

- A: 一个唯一的id

- B: 该方法指定的毫秒数
- C: 传递的函数
- D: `undefined`

▶ **答案**

# 43. 输出是什么?

```js
;[...'Lydia']
```

- A: `["L", "y", "d", "i", "a"]`
- B: `["Lydia"]`
- C: `[[], "Lydia"]`
- D: `[["L", "y", "d", "i", "a"]]`

▶ **答案**

# 44. 输出是什么?

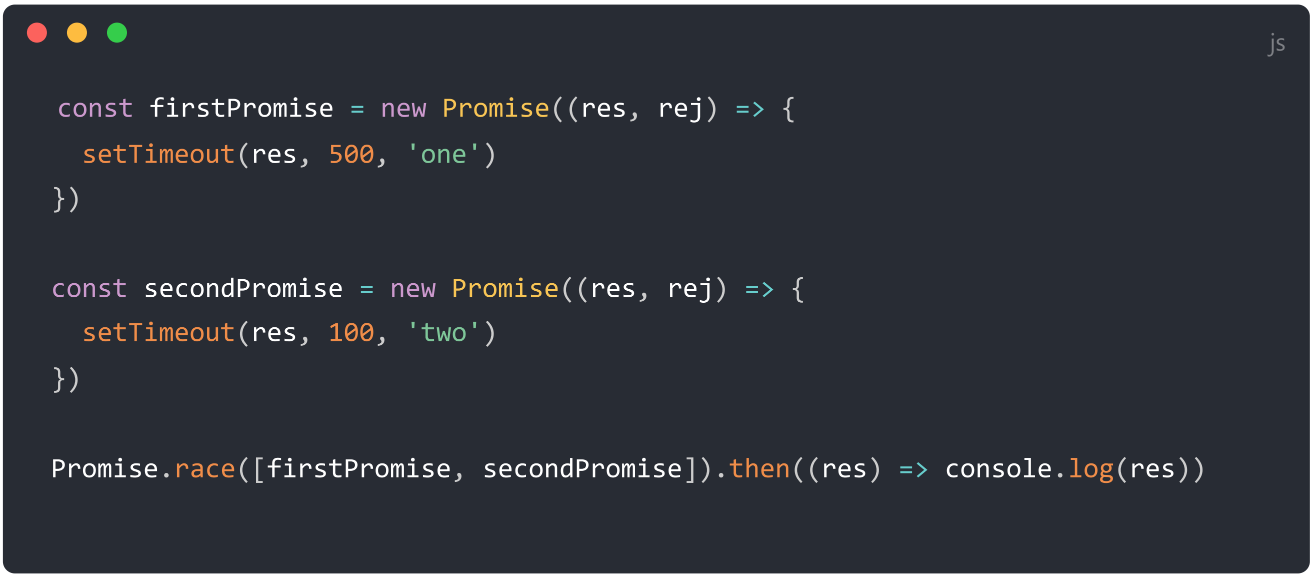
```js
function* generator(i) {
  yield i
  yield i * 2
}

const gen = generator(10)

console.log(gen.next().value)
console.log(gen.next().value)
```

- A: `[0, 10], [10, 20]`
- B: `20, 20`
- C: `10, 20`

- D: `0, 10 and 10, 20`

▶ 答案

# 45. 返回值是什么?

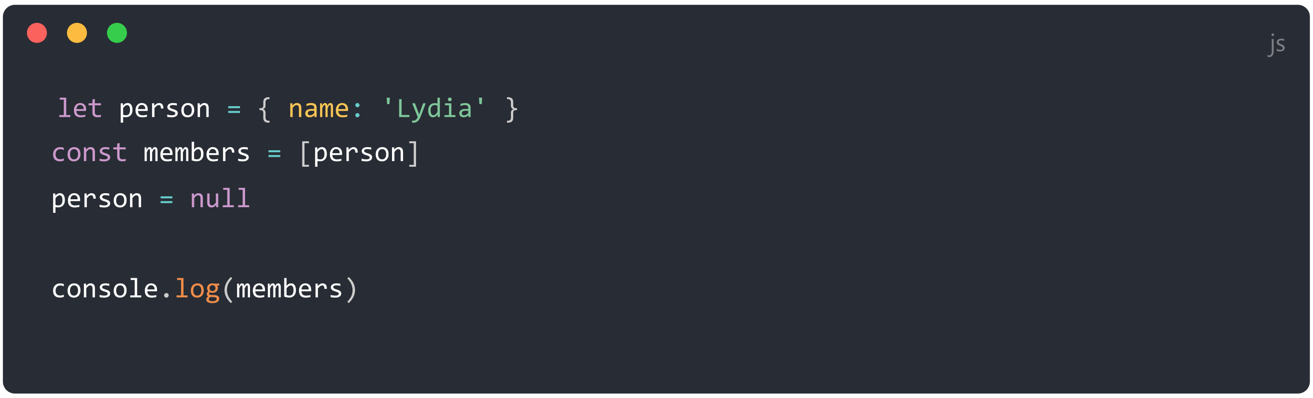


```js
const firstPromise = new Promise((res, rej) => {
  setTimeout(res, 500, 'one')
})

const secondPromise = new Promise((res, rej) => {
  setTimeout(res, 100, 'two')
})

Promise.race([firstPromise, secondPromise]).then((res) => console.log(res))
```

- A: `"one"`
- B: `"two"`
- C: `"two" "one"`
- D: `"one" "two"`

▶ 答案

# 46. 输出是什么?



```js
let person = { name: 'Lydia' }
const members = [person]
person = null

console.log(members)
```

- A: `null`
- B: `[null]`

- C: `[{}]`
- D: `[{ name: "Lydia" }]`

▶ 答案

# 47. 输出是什么?

```js
const person = {
  name: 'Lydia',
  age: 21
}

for (const item in person) {
  console.log(item)
}
```

- A: `{ name: "Lydia" }, { age: 21 }`
- B: `"name", "age"`
- C: `"Lydia", 21`
- D: `["name", "Lydia"], ["age", 21]`

▶ 答案

# 48. 输出是什么?

```js
console.log(3 + 4 + '5')
```

- A: `"345"`
- B: `"75"`
- C: `12`
- D: `"12"`

▶ **答案**

# 49. `num` 的值是什么?

```js
const num = parseInt('7*6', 10)
```

- A: `42`
- B: `"42"`
- C: `7`
- D: `NaN`

▶ **答案**

# 50. 输出是什么?

```js
;[1, 2, 3].map((num) => {
  if (typeof num === 'number') return
  return num * 2
})
```

- A: `[]`
- B: `[null, null, null]`
- C: `[undefined, undefined, undefined]`
- D: `[ 3 x empty ]`

▶ **答案**

# 51. 输出的是什么?

```js
 function getInfo(member, year) {
   member.name = 'Lydia'
   year = '1998'
 }

const person = { name: 'Sarah' }
const birthYear = '1997'

getInfo(person, birthYear)

console.log(person, birthYear)
```

- A: `{ name: "Lydia" }, "1997"`
- B: `{ name: "Sarah" }, "1998"`
- C: `{ name: "Lydia" }, "1998"`
- D: `{ name: "Sarah" }, "1997"`

▶ **答案**

# 52. 输出是什么?

```js
 function greeting() {
   throw 'Hello world!'
 }

function sayHi() {
  try {
    const data = greeting()
    console.log('It worked!', data)
  } catch (e) {
    console.log('Oh no an error!', e)
  }
}

sayHi()
```

- A: `"It worked! Hello world!"`
- B: `"Oh no an error: undefined`
- C: `SyntaxError: can only throw Error objects`
- D: `"Oh no an error: Hello world!`

▶ **答案**

# 53. 输出是什么?

```js
function Car() {
  this.make = 'Lamborghini'
  return { make: 'Maserati' }
}

const myCar = new Car()
console.log(myCar.make)
```

- A: `"Lamborghini"`
- B: `"Maserati"`
- C: `ReferenceError`
- D: `TypeError`

▶ **答案**

# 54. 输出是什么?

```js
;(() => {
  let x = (y = 10)
})()

console.log(typeof x)
console.log(typeof y)
```

- A: `"undefined", "number"`
- B: `"number", "number"`
- C: `"object", "number"`
- D: `"number", "undefined"`

▶ **答案**

# 55. 输出是什么?

```js
class Dog {
  constructor(name) {
    this.name = name
  }
}

Dog.prototype.bark = function () {
  console.log(`Woof I am ${this.name}`)
}

const pet = new Dog('Mara')

pet.bark()

delete Dog.prototype.bark

pet.bark()
```

- A: `"Woof I am Mara"`, `TypeError`
- B: `"Woof I am Mara"`, `"Woof I am Mara"`
- C: `"Woof I am Mara"`, `undefined`
- D: `TypeError`, `TypeError`

▶ **答案**

# 56. 输出是什么?

```js
const set = new Set([1, 1, 2, 3, 4])

console.log(set)
```

- A: `[1, 1, 2, 3, 4]`
- B: `[1, 2, 3, 4]`
- C: `{1, 1, 2, 3, 4}`
- D: `{1, 2, 3, 4}`

▶ **答案**

# 57. 输出是什么?

```js
// counter.js
let counter = 10
export default counter
```

```js
// index.js
import myCounter from './counter'

myCounter += 1

console.log(myCounter)
```

- A: `10`
- B: `11`
- C: `Error`
- D: `NaN`

▶ **答案**

# 58. 输出是什么?

```js
 const name = 'Lydia'
age = 21

console.log(delete name)
console.log(delete age)
```

- A: `false`, `true`
- B: `"Lydia"`, `21`
- C: `true`, `true`
- D: `undefined`, `undefined`

▶ **答案**

# 59. 输出是什么?

```js
 const numbers = [1, 2, 3, 4, 5]
const [y] = numbers

console.log(y)
```

- A: `[[1, 2, 3, 4, 5]]`
- B: `[1, 2, 3, 4, 5]`
- C: `1`
- D: `[1]`

▶ **答案**

# 60. 输出是什么?

```js
const user = { name: 'Lydia', age: 21 }
const admin = { admin: true, ...user }

console.log(admin)
```

- A: `{ admin: true, user: { name: "Lydia", age: 21 } }`
- B: `{ admin: true, name: "Lydia", age: 21 }`
- C: `{ admin: true, user: ["Lydia", 21] }`
- D: `{ admin: true }`

▶ **答案**

# 61. 输出是什么?

```js
const person = { name: 'Lydia' }

Object.defineProperty(person, 'age', { value: 21 })

console.log(person)
console.log(Object.keys(person))
```

- A: `{ name: "Lydia", age: 21 }`, `["name", "age"]`
- B: `{ name: "Lydia", age: 21 }`, `["name"]`
- C: `{ name: "Lydia"}`, `["name", "age"]`
- D: `{ name: "Lydia"}`, `["age"]`

▶ **答案**

# 62. 输出是什么?

```js
const settings = {
  username: 'lydiahallie',
  level: 19,
  health: 90
}

const data = JSON.stringify(settings, ['level', 'health'])
console.log(data)
```

- A: `"{"level":19, "health":90}"`
- B: `"{"username": "lydiahallie"}"`
- C: `"["level", "health"]"`
- D: `"{"username": "lydiahallie", "level":19, "health":90}"`

▶ **答案**

# 63. 输出是什么?

```js
let num = 10

const increaseNumber = () => num++
const increasePassedNumber = (number) => number++

const num1 = increaseNumber()
const num2 = increasePassedNumber(num1)

console.log(num1)
console.log(num2)
```

- A: `10` , `10`

- B: `10` , `11`
- C: `11` , `11`
- D: `11` , `12`

▶ **答案**

# 64. 输出什么?

```js
const value = { number: 10 }

const multiply = (x = { ...value }) => {
  console.log((x.number *= 2))
}

multiply()
multiply()
multiply(value)
multiply(value)
```

- A: `20` , `40` , `80` , `160`
- B: `20` , `40` , `20` , `40`
- C: `20` , `20` , `20` , `40`
- D: `NaN` , `NaN` , `20` , `40`

▶ **答案**

# 65. 输出什么?

```js
;[1, 2, 3, 4].reduce((x, y) => console.log(x, y))
```

- A: `1` `2` and `3` `3` and `6` `4`
- B: `1` `2` and `2` `3` and `3` `4`

- C: `1` `undefined` and `2` `undefined` and `3` `undefined` and `4` `undefined`
- D: `1` `2` and `undefined` `3` and `undefined` `4`

▶ **答案**

66. 使用哪个构造函数可以成功继承 `Dog` 类?

```js
class Dog {
  constructor(name) {
    this.name = name;
  }
};

class Labrador extends Dog {
  // 1
  constructor(name, size) {
    this.size = size;
  }
  // 2
  constructor(name, size) {
    super(name);
    this.size = size;
  }
  // 3
  constructor(size) {
    super(name);
    this.size = size;
  }
  // 4
  constructor(name, size) {
    this.name = name;
    this.size = size;
  }

};
```

- A: 1
- B: 2
- C: 3

- D: 4

▶ **答案**

# 67. 输出什么?

```js
// index.js
console.log('running index.js')
import { sum } from './sum.js'
console.log(sum(1, 2))

// sum.js
console.log('running sum.js')
export const sum = (a, b) => a + b
```

- A: `running index.js`, `running sum.js`, `3`
- B: `running sum.js`, `running index.js`, `3`
- C: `running sum.js`, `3`, `running index.js`
- D: `running index.js`, `undefined`, `running sum.js`

▶ **答案**

# 68. 输出什么?

```js
console.log(Number(2) === Number(2))
console.log(Boolean(false) === Boolean(false))
console.log(Symbol('foo') === Symbol('foo'))
```

- A: `true`, `true`, `false`
- B: `false`, `true`, `false`
- C: `true`, `false`, `true`
- D: `true`, `true`, `true`

▶ **答案**

# 69. 输出什么?

```js
 const name = 'Lydia Hallie'
console.log(name.padStart(13))
console.log(name.padStart(2))
```

- A: `"Lydia Hallie"`, `"Lydia Hallie"`
- B: `" Lydia Hallie"`, `" Lydia Hallie"` (`"[13x whitespace]Lydia Hallie"`, `"[2x whitespace]Lydia Hallie"`)
- C: `" Lydia Hallie"`, `"Lydia Hallie"` (`"[1x whitespace]Lydia Hallie"`, `"Lydia Hallie"`)
- D: `"Lydia Hallie"`, `"Lyd"`

▶ **答案**

# 70. 输出什么?

```js
console.log('🥑' + '💻')
```

- A: `"🥑💻"`
- B: `257548`
- C: A string containing their code points
- D: Error

▶ **答案**

# 71. 如何能打印出 `console.log` 语句后注释掉的值?

```js
function* startGame() {
  const answer = yield 'Do you love JavaScript?'
  if (answer !== 'Yes') {
    return "Oh wow... Guess we're gone here"
  }
  return 'JavaScript loves you back ❤️'
}

const game = startGame()
console.log(/* 1 */) // Do you love JavaScript?
console.log(/* 2 */) // JavaScript loves you back ❤️
```

- A: `game.next("Yes").value` and `game.next().value`
- B: `game.next.value("Yes")` and `game.next.value()`
- C: `game.next().value` and `game.next("Yes").value`
- D: `game.next.value()` and `game.next.value("Yes")`

▶ **答案**

# 72. 输出什么?

```js
console.log(String.raw`Hello\nworld`)
```

- A: `Hello world!`
- B: `Hello`
  `world`
- C: `Hello\nworld`
- D: `Hello\n`
  `world`

▶ **答案**

# 73. 输出什么?

```js
async function getData() {
  return await Promise.resolve('I made it!')
}

const data = getData()
console.log(data)
```

- A: `"I made it!"`
- B: `Promise {<resolved>: "I made it!"}`
- C: `Promise {<pending>}`
- D: `undefined`

▶ **答案**

# 74. 输出什么?

```js
function addToList(item, list) {
  return list.push(item)
}

const result = addToList('apple', ['banana'])
console.log(result)
```

- A: `['apple', 'banana']`
- B: `2`
- C: `true`
- D: `undefined`

▶ **答案**

# 75. 输出什么?

```js
const box = { x: 10, y: 20 }

Object.freeze(box)

const shape = box
shape.x = 100
console.log(shape)
```

- A: `{ x: 100, y: 20 }`
- B: `{ x: 10, y: 20 }`
- C: `{ x: 100 }`
- D: `ReferenceError`

▶ **答案**

# 76. 输出什么?

```js
const { name: myName } = { name: 'Lydia' }

console.log(name)
```

- A: `"Lydia"`
- B: `"myName"`
- C: `undefined`
- D: `ReferenceError`

▶ **答案**

# 77. 以下是个纯函数么?

```js
function sum(a, b) {
  return a + b
}
```

- A: Yes
- B: No

▶ **答案**

# 78. 输出什么?

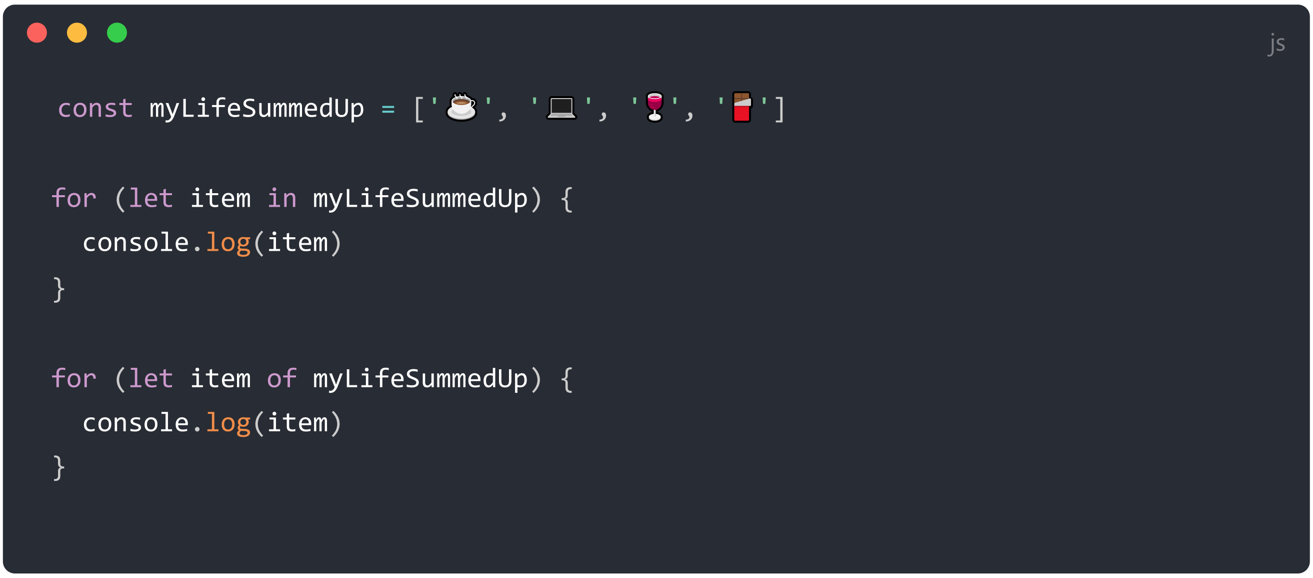
```js
const add = () => {
  const cache = {}
  return (num) => {
    if (num in cache) {
      return `From cache! ${cache[num]}`
    } else {
      const result = num + 10
      cache[num] = result
      return `Calculated! ${result}`
    }
  }
}

const addFunction = add()
console.log(addFunction(10))
console.log(addFunction(10))
console.log(addFunction(5 * 2))
```

- A: `Calculated! 20` `Calculated! 20` `Calculated! 20`
- B: `Calculated! 20` `From cache! 20` `Calculated! 20`

- C: `Calculated! 20` `From cache! 20` `From cache! 20`
- D: `Calculated! 20` `From cache! 20` `Error`

▶ 答案

# 79. 输出什么?

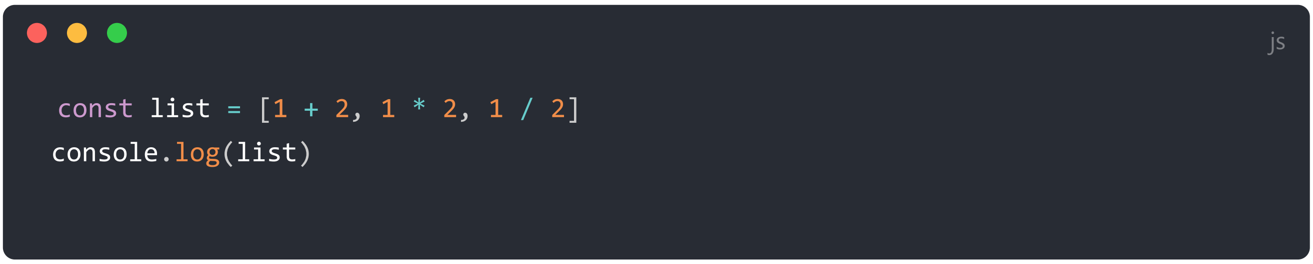


```js
 const myLifeSummedUp = ['☕', '💻', '🍷', '🍫']

for (let item in myLifeSummedUp) {
  console.log(item)
}

for (let item of myLifeSummedUp) {
  console.log(item)
}
```

- A: `0` `1` `2` `3` and `"☕"` `"💻"` `"🍷"` `"🍫"`
- B: `"☕"` `"💻"` `"🍷"` `"🍫"` and `"☕"` `"💻"` `"🍷"` `"🍫"`
- C: `"☕"` `"💻"` `"🍷"` `"🍫"` and `0` `1` `2` `3`
- D: `0` `1` `2` `3` and `{0: "☕", 1: "💻", 2: "🍷", 3: "🍫"}`

▶ 答案

# 80. 输出什么?



```js
 const list = [1 + 2, 1 * 2, 1 / 2]
console.log(list)
```

- A: `["1 + 2", "1 * 2", "1 / 2"]`
- B: `["12", 2, 0.5]`
- C: `[3, 2, 0.5]`

- D: `[1, 1, 1]`

▶ **答案**

# 81. 输出什么?

```js
function sayHi(name) {
  return `Hi there, ${name}`
}

console.log(sayHi())
```

- A: `Hi there,`
- B: `Hi there, undefined`
- C: `Hi there, null`
- D: `ReferenceError`

▶ **答案**

# 82. 输出什么?

```js
var status = '😎'

setTimeout(() => {
  const status = '😍'

  const data = {
    status: '🥑',
    getStatus() {
      return this.status
    }
  }

  console.log(data.getStatus())
  console.log(data.getStatus.call(this))
}, 0)
```

- A: `"🥑"` and `"😍"`
- B: `"🥑"` and `"😎"`
- C: `"😍"` and `"😎"`
- D: `"😎"` and `"😎"`

▶ **答案**

# 83. 输出什么?

```js
const person = {
  name: 'Lydia',
  age: 21
}

let city = person.city
city = 'Amsterdam'

console.log(person)
```

- A: `{ name: "Lydia", age: 21 }`
- B: `{ name: "Lydia", age: 21, city: "Amsterdam" }`
- C: `{ name: "Lydia", age: 21, city: undefined }`
- D: `"Amsterdam"`

▶ **答案**

# 84. 输出什么?

```js
function checkAge(age) {
  if (age < 18) {
    const message = "Sorry, you're too young."
  } else {
    const message = "Yay! You're old enough!"
  }

  return message
}

console.log(checkAge(21))
```

- A: `"Sorry, you're too young."`
- B: `"Yay! You're old enough!"`
- C: `ReferenceError`
- D: `undefined`

▶ **答案**

# 85. 什么样的信息将被打印?

```js
fetch('https://www.website.com/api/user/1')
  .then((res) => res.json())
  .then((res) => console.log(res))
```

- A: `fetch` 方法的结果
- B: 第二次调用 `fetch` 方法的结果
- C: 前一个 `.then()` 中回调方法返回的结果
- D: 总是 `undefined`

▶ **答案**

# 86. 哪个选项是将 `hasName` 设置为 `true` 的方法，前提是不能将 `true` 作为参数传递?

```js
function getName(name) {
  const hasName = //
}
```

- A: `!!name`
- B: `name`
- C: `new Boolean(name)`
- D: `name.length`

▶ **答案**

# 87. 输出什么?

```js
console.log('I want pizza'[0])
```

- A: `"""`
- B: `"I"`
- C: `SyntaxError`
- D: `undefined`

▶ **答案**

# 88. 输出什么?

```js
function sum(num1, num2 = num1) {
  console.log(num1 + num2)
}

sum(10)
```

- A: `NaN`
- B: `20`
- C: `ReferenceError`
- D: `undefined`

▶ **答案**

# 89. 输出什么?

```js
// module.js
export default () => 'Hello world'
export const name = 'Lydia'

// index.js
import * as data from './module'

console.log(data)
```

- A: `{ default: function default(), name: "Lydia" }`
- B: `{ default: function default() }`
- C: `{ default: "Hello world", name: "Lydia" }`
- D: Global object of `module.js`

▶ **答案**

# 90. 输出什么?

```js
class Person {
  constructor(name) {
    this.name = name
  }
}

const member = new Person('John')
console.log(typeof member)
```

- A: `"class"`
- B: `"function"`
- C: `"object"`
- D: `"string"`

▶ **答案**

# 91. 输出什么?

```js
let newList = [1, 2, 3].push(4)

console.log(newList.push(5))
```

- A: `[1, 2, 3, 4, 5]`
- B: `[1, 2, 3, 5]`
- C: `[1, 2, 3, 4]`
- D: `Error`

▶ **答案**

# 92. 输出什么?

```js
function giveLydiaPizza() {
  return 'Here is pizza!'
}

const giveLydiaChocolate = () => "Here's chocolate... now go hit the gym already."

console.log(giveLydiaPizza.prototype)
console.log(giveLydiaChocolate.prototype)
```

- A: `{ constructor: ...}` `{ constructor: ...}`
- B: `{}` `{ constructor: ...}`
- C: `{ constructor: ...}` `{}`
- D: `{ constructor: ...}` `undefined`

▶ **答案**

# 93. 输出什么?

```js
const person = {
  name: 'Lydia',
  age: 21
}

for (const [x, y] of Object.entries(person)) {
  console.log(x, y)
}
```

- A: `name` `Lydia` and `age` `21`
- B: `["name", "Lydia"]` and `["age", 21]`
- C: `["name", "age"]` and `undefined`
- D: `Error`

▶ **答案**

# 94. 输出什么?

```js
function getItems(fruitList, ...args, favoriteFruit) {
  return [...fruitList, ...args, favoriteFruit]
}

getItems(["banana", "apple"], "pear", "orange")
```

- A: `["banana", "apple", "pear", "orange"]`
- B: `[["banana", "apple"], "pear", "orange"]`
- C: `["banana", "apple", ["pear"], "orange"]`
- D: `SyntaxError`

▶ **答案**

# 95. 输出什么?

```js
function nums(a, b) {
  if (a > b) console.log('a is bigger')
  else console.log('b is bigger')
  return
  a + b
}

console.log(nums(4, 2))
console.log(nums(1, 2))
```

- A: `a is bigger`, `6` and `b is bigger`, `3`
- B: `a is bigger`, `undefined` and `b is bigger`, `undefined`
- C: `undefined` and `undefined`

- D: `SyntaxError`

▶ **答案**

# 96. 输出什么?

```js
class Person {
  constructor() {
    this.name = 'Lydia'
  }
}

Person = class AnotherPerson {
  constructor() {
    this.name = 'Sarah'
  }
}

const member = new Person()
console.log(member.name)
```

- A: `"Lydia"`
- B: `"Sarah"`
- C: `Error: cannot redeclare Person`
- D: `SyntaxError`

▶ **答案**

# 97. 输出什么?

```js
const info = {
  [Symbol('a')]: 'b'
}
```

```js
console.log(info)
console.log(Object.keys(info))
```

- A: `{Symbol('a'): 'b'}` and `["{Symbol('a')"]`
- B: `{}` and `[]`
- C: `{ a: "b" }` and `["a"]`
- D: `{Symbol('a'): 'b'}` and `[]`

▶ **答案**

# 98. 输出什么?

```js
const getList = ([x, ...y]) => [x, y]
const getUser = (user) => ({ name: user.name, age: user.age })

const list = [1, 2, 3, 4]
const user = { name: 'Lydia', age: 21 }

console.log(getList(list))
console.log(getUser(user))
```

- A: `[1, [2, 3, 4]]` and `undefined`
- B: `[1, [2, 3, 4]]` and `{ name: "Lydia", age: 21 }`
- C: `[1, 2, 3, 4]` and `{ name: "Lydia", age: 21 }`
- D: `Error` and `{ name: "Lydia", age: 21 }`

▶ **答案**

# 99. 输出什么?

```js
const name = 'Lydia'
```

```js
console.log(name())
```

- A: `SyntaxError`
- B: `ReferenceError`
- C: `TypeError`
- D: `undefined`

▶ **答案**

# 100. 输出什么?

```js
// 🎉✨ This is my 100th question! ✨🎉

const output = `${[] && 'Im'}possible!
You should${'' && `n't`} see a therapist after so much JavaScript lol`
```

- A: `possible! You should see a therapist after so much JavaScript lol`
- B: `Impossible! You should see a therapist after so much JavaScript lol`
- C: `possible! You shouldn't see a therapist after so much JavaScript lol`
- D: `Impossible! You shouldn't see a therapist after so much JavaScript lol`

▶ **答案**

# 101.输出什么?

```js
 const one = false || {} || null
const two = null || false || ''
const three = [] || 0 || true

console.log(one, two, three)
```

- A: `false` `null` `[]`
- B: `null` `""` `true`
- C: `{}` `""` `[]`
- D: `null` `null` `true`

▶ **答案**