JavaScript Questions

> [!NOTE]
> This repo was created in 2019 and the questions provided here are therefore based on the JavaScript syntax and behaviour at that time. Since JavaScript is a constantly evolving language, there are newer language features that are not covered by the questions here.

From basic to advanced: test how well you know JavaScript, refresh your knowledge a bit, or prepare for your coding interview! :muscle: :rocket: I update this repo regularly with new questions. I added the answers in the **collapsed sections** below the questions, simply click on them to expand it. It's just for fun, good luck! :heart:

Feel free to reach out to me! 😊

Instagram || Twitter || LinkedIn || Blog

Feel free to use them in a project! 😃 I would *really* appreciate a reference to this repo, I create the questions and explanations (yes I'm sad lol) and the community helps me so much to maintain and improve it! 💪 Thank you and have fun!

▶ **See 20 Available Translations** SAEGBADEESFRIDJPKRNLBRRUTHTRUAVNCNTWXK

1. What's the output?

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

sayHi();
```

- A: `Lydia` and `undefined`
- B: `Lydia` and `ReferenceError`

- C: `ReferenceError` and `21`
- D: `undefined` and `ReferenceError`

▶ **Answer**

2. What's the output?

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

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

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

▶ **Answer**

3. What's the output?

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

console.log(shape.diameter());
console.log(shape.perimeter());
```

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

▶ **Answer**

4. What's the output?

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

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

▶ **Answer**

5. Which one is true?

```
const bird = {
  size: 'small',
};

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

- A: `mouse.bird.size` is not valid
- B: `mouse[bird.size]` is not valid
- C: `mouse[bird["size"]]` is not valid
- D: All of them are valid

▶ **Answer**

6. What's the output?

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

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

- A: `Hello`
- B: `Hey!`
- C: `undefined`
- D: `ReferenceError`
- E: `TypeError`

▶ **Answer**

7. What's the output?

```
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`

▶ **Answer**

8. What's the output?

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

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

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

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

▶ **Answer**

9. What's the output?

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

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

▶ **Answer**

10. What happens when we do this?

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

bark.animal = 'dog';
```

- A: Nothing, this is totally fine!
- B: `SyntaxError` . You cannot add properties to a function this way.
- C: `"Woof"` gets logged.
- D: `ReferenceError`

▶ **Answer**

11. What's the output?

```
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`

▶ **Answer**

12. What's the output?

```
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`

▶ **Answer**

13. What are the three phases of event propagation?

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

▶ **Answer**

14. All object have prototypes.

- A: true
- B: false

▶ **Answer**

15. What's the output?

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

sum(1, '2');
```

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

▶ **Answer**

16. What's the output?

```
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`

▶ **Answer**

17. What's the output?

```
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`

▶ **Answer**

18. What's the output?

```
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`

▶ **Answer**

19. What's the output?

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

getAge(21);
```

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

▶ **Answer**

20. What's the output?

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

getAge();
```

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

▶ **Answer**

21. What's the value of `sum`?

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

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

▶ **Answer**

22. How long is cool_secret accessible?

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

- A: Forever, the data doesn't get lost.
- B: When the user closes the tab.
- C: When the user closes the entire browser, not only the tab.
- D: When the user shuts off their computer.

▶ **Answer**

23. What's the output?

```
var num = 8;
var num = 10;

console.log(num);
```

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

▶ **Answer**

24. What's the output?

```
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`

▶ **Answer**

25. What's the output?

```
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`

▶ **Answer**

26. The JavaScript global execution context creates two things for you: the global object, and the "this" keyword.

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

▶ **Answer**

27. What's the output?

```
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`

▶ **Answer**

28. What's the output?

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

const name = 'Lydia';

console.log(name.giveLydiaPizza())
```

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

▶ **Answer**

29. What's the output?

```
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`

▶ **Answer**

30. What's the output?

```
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`

▶ **Answer**

31. What is the event.target when clicking the button?

```
<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: An array of all nested elements.

▶ **Answer**

32. When you click the paragraph, what's the logged output?

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

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

▶ **Answer**

33. What's the output?

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

function sayHi(age) {
  return `${this.name} is ${age}`;
}

console.log(sayHi.call(person, 21));
console.log(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`

▶ **Answer**

34. What's the output?

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

console.log(typeof sayHi());
```

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

▶ **Answer**

35. Which of these values are falsy?

```
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

▶ **Answer**

36. What's the output?

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

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

▶ **Answer**

37. What's the output?

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

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

▶ **Answer**

38. What's the output?

```
(() => {
  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`

▶ **Answer**

39. Everything in JavaScript is either a...

- A: primitive or object
- B: function or object
- C: trick question! only objects
- D: number or object

▶ **Answer**

40. What's the output?

```
[[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]`

▶ **Answer**

41. What's the output?

```
!!null;
!!'';
!!1;
```

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

▶ **Answer**

42. What does the `setInterval` method return in the browser?

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

- A: a unique id
- B: the amount of milliseconds specified
- C: the passed function
- D: `undefined`

▶ **Answer**

43. What does this return?

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

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

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

▶ **Answer**

44. What's the output?

```
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`

▶ **Answer**

45. What does this return?

```
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"`

▶ **Answer**

46. What's the output?

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

console.log(members);
```

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

▶ **Answer**

47. What's the output?

```
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]`

▶ **Answer**

48. What's the output?

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

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

▶ **Answer**

49. What's the value of `num`?

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

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

▶ **Answer**

50. What's the output?

```
[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 ]`

▶ **Answer**

51. What's the output?

```
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"`

▶ **Answer**

52. What's the output?

```
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!`

▶ **Answer**

53. What's the output?

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

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

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

▶ **Answer**

54. What's the output?

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

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

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

▶ **Answer**

55. What's the output?

```
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`

▶ **Answer**

56. What's the output?

```
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}`

▶ **Answer**

57. What's the output?

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


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

myCounter += 1;

console.log(myCounter);
```

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

▶ **Answer**

58. What's the output?

```
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`

▶ **Answer**

59. What's the output?

```
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]`

▶ **Answer**

60. What's the output?

```
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 }`

▶ **Answer**

61. What's the output?

```
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"]`

▶ **Answer**

62. What's the output?

```
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}"`

▶ **Answer**

63. What's the output?

```
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`

▶ **Answer**

64. What's the output?

```
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`

▶ **Answer**

65. What's the output?

```
[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`

▶ **Answer**

66. With which constructor can we successfully extend the `Dog` class?

```
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

▶ **Answer**

67. What's the output?

```
// 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`

▶ **Answer**

68. What's the output?

```
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`

▶ **Answer**

69. What's the output?

```
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"`,

▶ **Answer**

70. What's the output?

```
console.log('🖱️' + '💻');
```

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

▶ **Answer**

71. How can we log the values that are commented out after the console.log statement?

```
function* startGame() {
  const answer = yield 'Do you love JavaScript?';
  if (answer !== 'Yes') {
    return "Oh wow... Guess we're done 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")`

▶ **Answer**

72. What's the output?

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

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

▶ **Answer**

73. What's the output?

```
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`

▶ **Answer**

74. What's the output?

```
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`

▶ **Answer**

75. What's the output?

```
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`

▶ **Answer**

76. What's the output?

```
const { firstName: myName } = { firstName: 'Lydia' };

console.log(firstName);
```

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

▶ **Answer**

77. Is this a pure function?

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

- A: Yes
- B: No

▶ **Answer**

78. What is the output?

```
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`

▶ **Answer**

79. What is the output?

```
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: "🥫"}`

▶ **Answer**

80. What is the output?

```
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]`

▶ **Answer**

81. What is the output?

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

console.log(sayHi());
```

- A: `Hi there,`

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

▶ **Answer**

82. What is the output?

```
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 "😎"

▶ **Answer**

83. What is the output?

```
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"`

▶ **Answer**

84. What is the output?

```
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`

▶ **Answer**

85. What kind of information would get logged?

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

- A: The result of the `fetch` method.
- B: The result of the second invocation of the `fetch` method.
- C: The result of the callback in the previous `.then()`.
- D: It would always be undefined.

▶ **Answer**

86. Which option is a way to set `hasName` equal to `true`, provided you cannot pass `true` as an argument?

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

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

▶ **Answer**

87. What's the output?

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

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

▶ **Answer**

88. What's the output?

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

sum(10);
```

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

▶ **Answer**

89. What's the output?

```
// 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`

▶ **Answer**

90. What's the output?

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

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

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

▶ **Answer**

91. What's the output?

```
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`

▶ **Answer**

92. What's the output?

```
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`

▶ **Answer**

93. What's the output?

```
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`

▶ **Answer**

94. What's the output?

```
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`

▶ **Answer**

95. What's the output?

```
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`

▶ **Answer**

96. What's the output?

```
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`

▶ **Answer**

97. What's the output?

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

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 `[]`

▶ **Answer**

98. What's the output?

```
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 `SyntaxError`
- 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 }`

▶ **Answer**

99. What's the output?

```
const name = 'Lydia';

console.log(name());
```

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

▶ **Answer**

100. What's the value of output?

```
// 🎉✨ 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

▶ **Answer**

101. What's the value of output?

```
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`

▶ **Answer**

102. What's the value of output?

```
const myPromise = () => Promise.resolve('I have resolved!');

function firstFunction() {
  myPromise().then(res => console.log(res));
  console.log('second');
}

async function secondFunction() {
  console.log(await myPromise());
  console.log('second');
}

firstFunction();
secondFunction();
```

- A: `I have resolved!`, `second` and `I have resolved!`, `second`
- B: `second`, `I have resolved!` and `second`, `I have resolved!`
- C: `I have resolved!`, `second` and `second`, `I have resolved!`

- D: `second`, `I have resolved!` and `I have resolved!`, `second`

▶ **Answer**

103. What's the value of output?

```
const set = new Set();

set.add(1);
set.add('Lydia');
set.add({ name: 'Lydia' });

for (let item of set) {
  console.log(item + 2);
}
```

- A: `3`, `NaN`, `NaN`
- B: `3`, `7`, `NaN`
- C: `3`, `Lydia2`, `[object Object]2`
- D: `"12"`, `Lydia2`, `[object Object]2`

▶ **Answer**

104. What's its value?

```
Promise.resolve(5);
```

- A: `5`
- B: `Promise {<pending>: 5}`
- C: `Promise {<fulfilled>: 5}`
- D: `Error`

▶ **Answer**

105. What's its value?

```
function compareMembers(person1, person2 = person) {
  if (person1 !== person2) {
    console.log('Not the same!');
  } else {
    console.log('They are the same!');
  }
}

const person = { name: 'Lydia' };

compareMembers(person);
```

- A: `Not the same!`
- B: `They are the same!`
- C: `ReferenceError`
- D: `SyntaxError`

▶ **Answer**

106. What's its value?

```
const colorConfig = {
  red: true,
  blue: false,
  green: true,
  black: true,
  yellow: false,
};

const colors = ['pink', 'red', 'blue'];

console.log(colorConfig.colors[1]);
```

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

▶ **Answer**

107. What's its value?

```
console.log('❤' === '❤');
```

- A: `true`
- B: `false`

▶ **Answer**

108. Which of these methods modifies the original array?

```
const emojis = ['✨', '🔮', '🤩'];

emojis.map(x => x + '✨');
emojis.filter(x => x !== '🔮');
emojis.find(x => x !== '🔮');
emojis.reduce((acc, cur) => acc + '✨');
emojis.slice(1, 2, '✨');
emojis.splice(1, 2, '✨');
```

- A: `All of them`
- B: `map reduce slice splice`
- C: `map slice splice`
- D: `splice`

▶ **Answer**

109. What's the output?

```
const food = ['🍕', '🥤', '🔮', '🍔'];
const info = { favoriteFood: food[0] };

info.favoriteFood = '🍝';

console.log(food);
```

- A: `['🍕', '🥤', '🔮', '🍔']`
- B: `['🍝', '🥤', '🔮', '🍔']`
- C: `['🍝', '🍕', '🥤', '🔮', '🍔']`
- D: `ReferenceError`

▶ **Answer**

110. What does this method do?

```
JSON.parse();
```

- A: Parses JSON to a JavaScript value
- B: Parses a JavaScript object to JSON
- C: Parses any JavaScript value to JSON
- D: Parses JSON to a JavaScript object only

▶ **Answer**

111. What's the output?

```
let name = 'Lydia';

function getName() {
  console.log(name);
  let name = 'Sarah';
}

getName();
```

- A: Lydia
- B: Sarah
- C: `undefined`
- D: `ReferenceError`

▶ **Answer**

112. What's the output?

```
function* generatorOne() {
  yield ['a', 'b', 'c'];
}

function* generatorTwo() {
  yield* ['a', 'b', 'c'];
}

const one = generatorOne();
const two = generatorTwo();

console.log(one.next().value);
console.log(two.next().value);
```

- A: `a` and `a`
- B: `a` and `undefined`
- C: `['a', 'b', 'c']` and `a`
- D: `a` and `['a', 'b', 'c']`

▶ **Answer**

113. What's the output?

```
console.log(`${(x => x)('I love')} to program`);
```

- A: `I love to program`
- B: `undefined to program`
- C: `${(x => x)('I love') to program`
- D: `TypeError`

▶ **Answer**

114. What will happen?

```
let config = {
  alert: setInterval(() => {
    console.log('Alert!');
  }, 1000),
};

config = null;
```

- A: The `setInterval` callback won't be invoked
- B: The `setInterval` callback gets invoked once
- C: The `setInterval` callback will still be called every second
- D: We never invoked `config.alert()`, config is `null`

▶ **Answer**

115. Which method(s) will return the value `'Hello world!'`?

```
const myMap = new Map();
const myFunc = () => 'greeting';

myMap.set(myFunc, 'Hello world!');

//1
myMap.get('greeting');
//2
myMap.get(myFunc);
//3
myMap.get(() => 'greeting');
```

- A: 1
- B: 2
- C: 2 and 3
- D: All of them

▶ **Answer**

116. What's the output?

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

const changeAge = (x = { ...person }) => (x.age += 1);
const changeAgeAndName = (x = { ...person }) => {
  x.age += 1;
  x.name = 'Sarah';
};

changeAge(person);
changeAgeAndName();

console.log(person);
```

- A: `{name: "Sarah", age: 22}`
- B: `{name: "Sarah", age: 23}`
- C: `{name: "Lydia", age: 22}`
- D: `{name: "Lydia", age: 23}`

▶ **Answer**

117. Which of the following options will return `6` ?

```
function sumValues(x, y, z) {
  return x + y + z;
}
```

- A: `sumValues([...1, 2, 3])`
- B: `sumValues([...[1, 2, 3]])`
- C: `sumValues(...[1, 2, 3])`
- D: `sumValues([1, 2, 3])`

▶ **Answer**

118. What's the output?

```
let num = 1;
const list = ['🥳', '🤠', '😇', '🤪'];

console.log(list[(num += 1)]);
```

- A: 🤠
- B: 😇
- C: `SyntaxError`
- D: `ReferenceError`

▶ **Answer**

119. What's the output?

```
const person = {
  firstName: 'Lydia',
  lastName: 'Hallie',
  pet: {
    name: 'Mara',
    breed: 'Dutch Tulip Hound',
  },
  getFullName() {
    return `${this.firstName} ${this.lastName}`;
  },
};

console.log(person.pet?.name);
console.log(person.pet?.family?.name);
console.log(person.getFullName?.());
console.log(member.getLastName?.());
```

- A: `undefined undefined undefined undefined`
- B: `Mara undefined Lydia Hallie ReferenceError`
- C: `Mara null Lydia Hallie null`
- D: `null ReferenceError null ReferenceError`

▶ **Answer**

120. What's the output?

```
const groceries = ['banana', 'apple', 'peanuts'];

if (groceries.indexOf('banana')) {
  console.log('We have to buy bananas!');
} else {
  console.log(`We don't have to buy bananas!`);
}
```

- A: We have to buy bananas!
- B: We don't have to buy bananas
- C: `undefined`
- D: `1`

▶ **Answer**

121. What's the output?

```
const config = {
  languages: [],
  set language(lang) {
    return this.languages.push(lang);
  },
};

console.log(config.language);
```

- A: `function language(lang) { this.languages.push(lang }`
- B: `0`
- C: `[]`
- D: `undefined`

▶ **Answer**

122. What's the output?

```
const name = 'Lydia Hallie';

console.log(!typeof name === 'object');
console.log(!typeof name === 'string');
```

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

▶ **Answer**

123. What's the output?

```
const add = x => y => z => {
  console.log(x, y, z);
  return x + y + z;
};

add(4)(5)(6);
```

- A: `4  5  6`
- B: `6  5  4`
- C: `4  function  function`
- D: `undefined  undefined  6`

▶ **Answer**

124. What's the output?

```
async function* range(start, end) {
  for (let i = start; i <= end; i++) {
    yield Promise.resolve(i);
  }
}

(async () => {
  const gen = range(1, 3);
  for await (const item of gen) {
    console.log(item);
  }
})();
```

- A: `Promise {1}  Promise {2}  Promise {3}`
- B: `Promise {<pending>}  Promise {<pending>}  Promise {<pending>}`
- C: `1  2  3`
- D: `undefined  undefined  undefined`

▶ **Answer**

125. What's the output?

```
const myFunc = ({ x, y, z }) => {
  console.log(x, y, z);
};

myFunc(1, 2, 3);
```

- A: `1  2  3`
- B: `{1: 1}  {2: 2}  {3: 3}`
- C: `{ 1: undefined }  undefined  undefined`
- D: `undefined  undefined  undefined`

▶ **Answer**

126. What's the output?

```
function getFine(speed, amount) {
  const formattedSpeed = new Intl.NumberFormat('en-US', {
    style: 'unit',
    unit: 'mile-per-hour'
  }).format(speed);

  const formattedAmount = new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency: 'USD'
  }).format(amount);

  return `The driver drove ${formattedSpeed} and has to pay ${formattedAmount}`;
}

console.log(getFine(130, 300))
```

- A: The driver drove 130 and has to pay 300
- B: The driver drove 130 mph and has to pay $300.00
- C: The driver drove undefined and has to pay undefined
- D: The driver drove 130.00 and has to pay 300.00

▶ **Answer**

127. What's the output?

```
const spookyItems = ['👻', '🎃', '🕸'];
({ item: spookyItems[3] } = { item: '💀' });

console.log(spookyItems);
```

- A: ["👻", "🎃", "🕸"]
- B: ["👻", "🎃", "🕸", "💀"]
- C: ["👻", "🎃", "🕸", { item: "💀" }]
- D: ["👻", "🎃", "🕸", "[object Object]"]

▶ **Answer**

128. What's the output?

```
const name = 'Lydia Hallie';
const age = 21;

console.log(Number.isNaN(name));
console.log(Number.isNaN(age));

console.log(isNaN(name));
console.log(isNaN(age));
```

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

▶ **Answer**

129. What's the output?

```
const randomValue = 21;

function getInfo() {
  console.log(typeof randomValue);
  const randomValue = 'Lydia Hallie';
}

getInfo();
```

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

▶ **Answer**

130. What's the output?

```
const myPromise = Promise.resolve('Woah some cool data');

(async () => {
  try {
    console.log(await myPromise);
  } catch {
    throw new Error(`Oops didn't work`);
  } finally {
    console.log('Oh finally!');
  }
})();
```

- A: `Woah some cool data`
- B: `Oh finally!`
- C: `Woah some cool data  Oh finally!`
- D: `Oops didn't work  Oh finally!`

▶ **Answer**

131. What's the output?

```
const emojis = ['🥑', ['✨', '✨', ['🍕', '🍕']]];

console.log(emojis.flat(1));
```

- A: `['🥑', ['✨', '✨', ['🍕', '🍕']]]`
- B: `['🥑', '✨', '✨', ['🍕', '🍕']]`
- C: `['🥑', ['✨', '✨', '🍕', '🍕']]`
- D: `['🥑', '✨', '✨', '🍕', '🍕']`

▶ **Answer**

132. What's the output?

```
class Counter {
  constructor() {
    this.count = 0;
  }

  increment() {
    this.count++;
  }
}

const counterOne = new Counter();
counterOne.increment();
counterOne.increment();

const counterTwo = counterOne;
counterTwo.increment();

console.log(counterOne.count);
```

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

▶ **Answer**

133. What's the output?

```
const myPromise = Promise.resolve(Promise.resolve('Promise'));

function funcOne() {
  setTimeout(() => console.log('Timeout 1!'), 0);
  myPromise.then(res => res).then(res => console.log(`${res} 1!`));
  console.log('Last line 1!');
}

async function funcTwo() {
  const res = await myPromise;
  console.log(`${res} 2!`)
  setTimeout(() => console.log('Timeout 2!'), 0);
  console.log('Last line 2!');
}

funcOne();
funcTwo();
```

- A: `Promise 1! Last line 1! Promise 2! Last line 2! Timeout 1! Timeout 2!`
- B: `Last line 1! Timeout 1! Promise 1! Last line 2! Promise2! Timeout 2!`
- C: `Last line 1! Promise 2! Last line 2! Promise 1! Timeout 1! Timeout 2!`
- D: `Timeout 1! Promise 1! Last line 1! Promise 2! Timeout 2! Last line 2!`

▶ **Answer**

134. How can we invoke `sum` in `sum.js` from `index.js`?

```
// sum.js
export default function sum(x) {
  return x + x;
}

// index.js
import * as sum from './sum';
```

- A: `sum(4)`
- B: `sum.sum(4)`
- C: `sum.default(4)`
- D: Default aren't imported with `*`, only named exports

▶ **Answer**

135. What's the output?

```
const handler = {
  set: () => console.log('Added a new property!'),
  get: () => console.log('Accessed a property!'),
};

const person = new Proxy({}, handler);

person.name = 'Lydia';
person.name;
```

- A: `Added a new property!`
- B: `Accessed a property!`
- C: `Added a new property!  Accessed a property!`
- D: Nothing gets logged

▶ **Answer**

136. Which of the following will modify the `person` object?

```
const person = { name: 'Lydia Hallie' };

Object.seal(person);
```

- A: `person.name = "Evan Bacon"`
- B: `person.age = 21`
- C: `delete person.name`
- D: `Object.assign(person, { age: 21 })`

▶ **Answer**

137. Which of the following will modify the `person` object?

```
const person = {
  name: 'Lydia Hallie',
  address: {
    street: '100 Main St',
  },
};

Object.freeze(person);
```

- A: `person.name = "Evan Bacon"`
- B: `delete person.address`
- C: `person.address.street = "101 Main St"`
- D: `person.pet = { name: "Mara" }`

▶ **Answer**

138. What's the output?

```
const add = x => x + x;

function myFunc(num = 2, value = add(num)) {
  console.log(num, value);
}

myFunc();
myFunc(3);
```

- A: `2 4` and `3 6`
- B: `2 NaN` and `3 NaN`
- C: `2 Error` and `3 6`
- D: `2 4` and `3 Error`

▶ **Answer**

139. What's the output?

```
class Counter {
  #number = 10

  increment() {
    this.#number++
  }

  getNum() {
    return this.#number
  }
}

const counter = new Counter()
counter.increment()

console.log(counter.#number)
```

- A: `10`
- B: `11`
- C: `undefined`
- D: `SyntaxError`

▶ **Answer**

140. What's missing?

```
const teams = [
  { name: 'Team 1', members: ['Paul', 'Lisa'] },
  { name: 'Team 2', members: ['Laura', 'Tim'] },
];

function* getMembers(members) {
  for (let i = 0; i < members.length; i++) {
    yield members[i];
  }
}

function* getTeams(teams) {
  for (let i = 0; i < teams.length; i++) {
    // ✨ SOMETHING IS MISSING HERE ✨
  }
}

const obj = getTeams(teams);
obj.next(); // { value: "Paul", done: false }
obj.next(); // { value: "Lisa", done: false }
```

- A: `yield getMembers(teams[i].members)`
- B: `yield* getMembers(teams[i].members)`
- C: `return getMembers(teams[i].members)`
- D: `return yield getMembers(teams[i].members)`

▶ **Answer**

141. What's the output?

```
const person = {
  name: 'Lydia Hallie',
  hobbies: ['coding'],
};

function addHobby(hobby, hobbies = person.hobbies) {
  hobbies.push(hobby);
  return hobbies;
}

addHobby('running', []);
addHobby('dancing');
addHobby('baking', person.hobbies);

console.log(person.hobbies);
```

- A: `["coding"]`
- B: `["coding", "dancing"]`
- C: `["coding", "dancing", "baking"]`
- D: `["coding", "running", "dancing", "baking"]`

▶ **Answer**

142. What's the output?

```
class Bird {
  constructor() {
    console.log("I'm a bird. 🦢");
  }
}

class Flamingo extends Bird {
  constructor() {
    console.log("I'm pink. 🌸");
    super();
  }
}

const pet = new Flamingo();
```

- A: `I'm pink. 🌸`

- B: `I'm pink. 🌸 I'm a bird. 🦢`
- C: `I'm a bird. 🦢 I'm pink. 🌸`
- D: Nothing, we didn't call any method

▶ **Answer**

143. Which of the options result(s) in an error?

```
const emojis = ['🎄', '👼', '🎁', '⭐'];

/* 1 */ emojis.push('🦌');
/* 2 */ emojis.splice(0, 2);
/* 3 */ emojis = [...emojis, '🥂'];
/* 4 */ emojis.length = 0;
```

- A: 1
- B: 1 and 2
- C: 3 and 4
- D: 3

▶ **Answer**

144. What do we need to add to the `person` object to get `["Lydia Hallie", 21]` as the output of `[...person]`?

```
const person = {
  name: "Lydia Hallie",
  age: 21
}

[...person] // ["Lydia Hallie", 21]
```

- A: Nothing, object are iterable by default
- B: `*[Symbol.iterator]() { for (let x in this) yield* this[x] }`
- C: `*[Symbol.iterator]() { yield* Object.values(this) }`
- D: `*[Symbol.iterator]() { for (let x in this) yield this }`

▶ **Answer**

145. What's the output?

```
let count = 0;
const nums = [0, 1, 2, 3];

nums.forEach(num => {
        if (num) count += 1
})

console.log(count)
```

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

▶ **Answer**

146. What's the output?

```
function getFruit(fruits) {
        console.log(fruits?.[1]?.[1])
}

getFruit([['🍊', '🍇'], ['🍍']])
getFruit()
getFruit([['🍍'], ['🍊', '🍇']])
```

- A: `null` , `undefined` , 🍇
- B: `[]` , `null` , 🍇
- C: `[]` , `[]` , 🍇
- D: `undefined` , `undefined` , 🍇

▶ **Answer**

147. What's the output?

```
class Calc {
        constructor() {
                this.count = 0
        }

        increase() {
                this.count ++
        }
}

const calc = new Calc()
new Calc().increase()

console.log(calc.count)
```

- A: `0`
- B: `1`
- C: `undefined`
- D: `ReferenceError`

▶ **Answer**

148. What's the output?

```
const user = {
        email: "e@mail.com",
        password: "12345"
}

const updateUser = ({ email, password }) => {
        if (email) {
                Object.assign(user, { email })
        }

        if (password) {
                user.password = password
        }

        return user
}

const updatedUser = updateUser({ email: "new@email.com" })

console.log(updatedUser === user)
```

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

▶ **Answer**

149. What's the output?

```
const fruit = ['🍌', '🍊', '🍐']

fruit.slice(0, 1)
fruit.splice(0, 1)
fruit.unshift('🍇')

console.log(fruit)
```

- A: `['🍌', '🍊', '🍐']`
- B: `['🍊', '🍐']`
- C: `['🍇', '🍊', '🍐']`

- D: ['🍇', '🦴', '🍈', '🍉']

▶ **Answer**

150. What's the output?

```
const animals = {};
let dog = { emoji: '🐶' }
let cat = { emoji: '🐈' }

animals[dog] = { ...dog, name: "Mara" }
animals[cat] = { ...cat, name: "Sara" }

console.log(animals[dog])
```

- A: `{ emoji: "🐶", name: "Mara" }`
- B: `{ emoji: "🐈", name: "Sara" }`
- C: `undefined`
- D: `ReferenceError`

▶ **Answer**

151. What's the output?

```
const user = {
        email: "my@email.com",
        updateEmail: email => {
                this.email = email
        }
}

user.updateEmail("new@email.com")
console.log(user.email)
```

- A: `my@email.com`
- B: `new@email.com`
- C: `undefined`
- D: `ReferenceError`

▶ **Answer**

152. What's the output?

```
const promise1 = Promise.resolve('First')
const promise2 = Promise.resolve('Second')
const promise3 = Promise.reject('Third')
const promise4 = Promise.resolve('Fourth')

const runPromises = async () => {
        const res1 = await Promise.all([promise1, promise2])
        const res2  = await Promise.all([promise3, promise4])
        return [res1, res2]
}

runPromises()
        .then(res => console.log(res))
        .catch(err => console.log(err))
```

- A: `[['First', 'Second'], ['Fourth']]`
- B: `[['First', 'Second'], ['Third', 'Fourth']]`
- C: `[['First', 'Second']]`
- D: `'Third'`

▶ **Answer**

153. What should the value of `method` be to log `{ name: "Lydia", age: 22 }`?

```
const keys = ["name", "age"]
const values = ["Lydia", 22]

const method = /* ?? */
Object[method](keys.map((_, i) => {
        return [keys[i], values[i]]
})) // { name: "Lydia", age: 22 }
```

- A: `entries`
- B: `values`
- C: `fromEntries`
- D: `forEach`

▶ **Answer**

154. What's the output?

```
const createMember = ({ email, address = {}}) => {
        const validEmail = /.+\@.+\..+/.test(email)
        if (!validEmail) throw new Error("Valid email pls")

        return {
                email,
                address: address ? address : null
        }
}

const member = createMember({ email: "my@email.com" })
console.log(member)
```

- A: `{ email: "my@email.com", address: null }`
- B: `{ email: "my@email.com" }`
- C: `{ email: "my@email.com", address: {} }`
- D: `{ email: "my@email.com", address: undefined }`

▶ **Answer**

155. What's the output?

```
let randomValue = { name: "Lydia" }
randomValue = 23

if (!typeof randomValue === "string") {
        console.log("It's not a string!")
} else {
        console.log("Yay it's a string!")
}
```

- A: `It's not a string!`
- B: `Yay it's a string!`
- C: `TypeError`
- D: `undefined`

▶ **Answer**