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`

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`

3. **What's the output?**

```
+true;
!"Lydia";
```

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

4. **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`

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

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`

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`

8. 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`

9. 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`

10.What's the output?

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

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

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

11. 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`

12.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`

13.What's the output?

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

getAge(21);
```

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

14.What's the output?

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

getAge();
```

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

15.What's value of `sum`?

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

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

16.What's the output?

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

console.log(num);
```

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

17.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`

18.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`

19.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`

20.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.

21.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`

22.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`

23.What's the output?

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

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

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

24.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

25.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`

26.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]`