JavaScript & React Interview Prep Summary

1. Var, Let, Const + Hoisting

- var: function scoped, hoisted and initialized with undefined.
- let & const: block scoped, hoisted but not initialized (Temporal Dead Zone).

Hoisting examples:

1) var example
```
console.log(a); // undefined
var a = 5;
```

2) let example
```
console.log(b); // ReferenceError
let b = 10;
```

3) function declaration
```
greet();
function greet() { console.log("Hello!"); } // Works
```

4) function expression (var)
```
greet();
var greet = function() { console.log("Hi!"); }; // TypeError
```

5) function expression (let/const)
```
greet();
const greet = () => { console.log("Hi!"); }; // ReferenceError
```

2. Arrow Functions & Hoisting

- Arrow functions are function expressions, not declarations.
- They behave like variables: hoisted but not initialized.
- Calling before definition leads to ReferenceError (let/const) or TypeError (var).

Summary Table:

Function Type	Hoisted?	Initialized Early?	Callable Before Definition?
function sayHi()	Yes	Yes	Yes

var sayHi = () => {} | Yes | No (undefined) | No (TypeError)
let/const sayHi = () => {}| Yes | No (TDZ) | No (ReferenceError)

3. Closures

Definition: A closure is when a function "remembers" variables from its outer scope even after the outer function has finished executing.

Example:

```
function outer() {
  let x = 10;
  function inner() {
    console.log(x);
  }
  return inner;
}
const closureFn = outer();
closureFn(); // 10
```

Real-life analogy: Carrying your lunchbox from home to school - you remember the lunch even outside your home.

Common use case: Private variables and data hiding

Example:

```
function counter() {
  let count = 0;
  return function () {
    count++;
    return count;
  };
}
const increment = counter();
console.log(increment()); // 1
```

```
console.log(increment()); // 2
```

Loop + Closure Example:

```
for (var i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), 1000);
} // prints 3,3,3 due to var scope

for (let i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), 1000);
} // prints 0,1,2 due to block scope
```

4. Terminology (Simple Terms)

- Closure: A backpack a function carries to remember old variables (scope memory).
- Scope: The room where a variable lives/is created.
- Temporal Dead Zone (TDZ): A danger zone where variables declared with let/const are hoisted but not yet accessible.
- Hoisting: JavaScript's magic trick of lifting declarations to the top before running the code.

5. Quiz Questions Recap

Q1: What does this print?

```
function greet() {
  const name = "Priyanshu";
  return function () { console.log("Hello, " + name); };
}
const sayHello = greet();
sayHello(); // Hello, Priyanshu
```

Q2: What is output?

```
function build() {
  let a = 100;
  return () => console.log(a);
}
let x = build();
```

JavaScript & React Interview Prep Summary

```
x(); // 100
```

Q3: What happens after timeout?

```
function magic() {
  const secret = "[magic wand]";
  return function () { console.log("Secret is: " + secret); };
}
const reveal = magic();
setTimeout(reveal, 5000); // Secret is: [magic wand]
```