Scope Quiz Recall

```
function letsJam() {
  // function1's scope
  let rand = 3;

  if (true) {
    const rand = 2;
  }

  if (true) {
    let rand = 1;
  }

  if (true) {
    const rand = "let's jam!";
  }

  return rand;
}

letsJam(); // ???
```

The value returned by the `letsJam` function is _.

- ◯ `3`
- ◯ `1`
- ◯ `let's jam!`
- ◯ An error is thrown
- ◯ `2`

EXPLANATION

The keywords `let` and `const` are block-scoped. Meaning that if a `let` or `const` are declared within a block `{}` that variable will stay within that block. In the above `letsJam` function the value returned will be the `rand` variable that was declared within the same outer scope - `3`.

```
function sayPuppy() {
  const puppy = "Wolfie";
  return puppy;
}

sayPuppy(); // "Wolfie"

console.log(puppy); // ????
```

What is the value logged in the last line of the snippet above (`console.log(puppy)`)?

- ◯ `puppy`
- ◯ `undefined`
- ◯ An Error is thrown
- ◯ `Wolfie`

EXPLANATION

Scope chaining allows an inner scope to reference an outer scope's variables but it will not allow an outer scope to access inner scope's variables.

```
function inner() {
  let str = "hello";
  return str;
}
```

```
function outer() {
  let test = inner();
  return test;
}

let result1 = outer();

result2 = inner();

result1 === result2; // ???
```

What is the value of the final line of the snippet above (`result1 === result2`) ?

- ◯ `true`
- ◯ `false`
- ◯ An Error is thrown

EXPLANATION

No matter where `inner` is invoked it will always return the same result. This is because of *lexical scoping*.

```
let puppy = "Shasta";

function sayPuppy() {
  console.log(puppy);
}

sayPuppy(); // ???
```

What is the value logged inside the `sayPuppy` function?

- ◯

- `puppy`
- `undefined`
- An Error is thrown
- `Shasta`

EXPLANATION

We declared a variable with `let` in the global scope. The `sayPuppy` function will have access to any variables within it's local scope as well as any variables declared in outer scopes because of scope chaining!

```
function catSound() {
  var sound = "meow";
  return sound;
}

function dogSound() {
  var sound = "bark";
  return sound;
}

let noise1 = catSound();
let noise2 = dogSound();

noise1 === noise2; // ???
```

The value of the last line in the code snippet above is: ____

- `false`
- `true`

EXPLANATION

Above we declared two different function-scoped variables using `var` in `catSound` and `dogSound`. Since the `var` declared variables will be function-scoped they will return different values from their separate functions.

```
// 1. ???
let chicken = "bokbok";

function farmTime() {
  // 2. ???
  console.log(chicken);

  if (true) {
    // 3. ???
    let cow = "moo";
  }
}
```

In the above code snippet there are three scopes labelled with numbers. Below pick the correct answer for the name of each scope in order.

- ◯ **1.**Local/Function Scope **2.**Global Scope **3.** Block Scope
- ◯ **1.** Global Scope **2.**Local/Function Scope **3.**Block Scope
- ◯ **1.** Block Scope **2.**Local/Function Scope **3.**Global Scope

EXPLANATION

The three scopes above are Global scope, Function scope, and Block scope in that order.