Predicting Variable Evaluations Quiz

```
let func1 = () => {
  let hello;
  console.log(hello);
};

let func2 = () => {
  console.log(hello);
  let hello;
};

let func3 = () => {
  console.log(hello);
  var hello;
};
```

Which of the above functions will throw an error when invoked?

- ◯ func2
- ◯ All three will throw errors
- ◯ func3
- ◯ func1

EXPLANATION

The `func1` function will run because a `let` declared variable with have a default value of `undefined` and will print that value. The `func3` function uses `var` to declare a variable which will hoist the name of the `hello` variable to the top of the function's scope - allowing it be logged with the default value of `undefined`. That leaves `func2` which will throw an error! This is because in `func2` we declare a variable using `let` which means that variable's name will be *hoisted* to the top of the function's scope but will be unavailable until it has been assigned a value because it is in the *temporal dead zone*.

```
const goodbye;
console.log(goodbye); // ???
```

What is printed when the above code snippet is run?

- ◯ `undefined`
- ◯ An Error is thrown.
- ◯ `goodbye`

EXPLANATION

When declaring a new `const` variable we need to assign that variable a value because of the nature of `const` being unable to be reassigned after the variable's declaration.

```
let goodbye;
console.log(goodbye); // ???
```

What is printed when the above code snippet is run?

- ◯ `goodbye`
- ◯ `undefined`
- ◯ An Error is thrown.

EXPLANATION

An declared but unassigned `let` variable will by default evaluate to `undefined`.

```
var hello;
console.log(hello); // ???
```

What is printed when the above code snippet is run?

- ◯ `hello`
- ◯ `undefined`
- ◯ An Error is thrown.

EXPLANATION

A declared but unassigned `var` variable will by default evaluate to `undefined`.