PART 1

LAB SESSION

- ✓ **Functions**

Lab 6a: Wrong insertion problem

1. Create *6a-wrong-insertion.js*

```
let functionA = () => {
    return {
        hello: "Hello"
    }
}
let functionB = () => {
    return
    {
        hello: "Hello"
    }
}
console.log(functionA() === functionB())

// What is the result?
```

What is the result?

Lab 6a: Wrong insertion problem

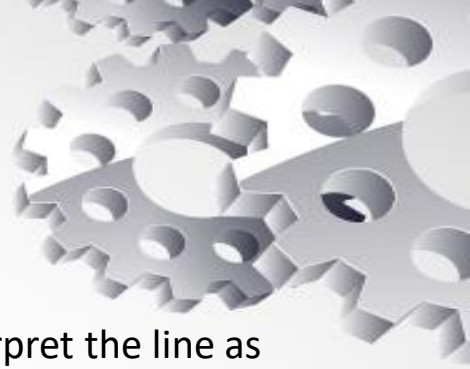
Answer:

The above expression will return false. This is the compiler interpreted the line ending with return keyword from functionB as the end of expression and implicitly inserted the semi-colon to the wrong place. This effectively becomes.

```
let functionB = () => {
    return;
    {
        hello: "Hello";
    }
};
```

functionB returns null and therefore is not equal to the result of functionA

Lab 6a: Wrong insertion problem



Always begin a return block after the return keyword

For this to work, the { should be added immediately after the return keyword so that the compiler does not interpret the line as the end of expression.

```
let child = () => {
    return {
        hello: "Hello",
    };
};
console.log(child());
// Returns { hello: 'Hello' }
```

Lab 6b: Wrong omission problem

1. Create *6b-wrong-omission.js*

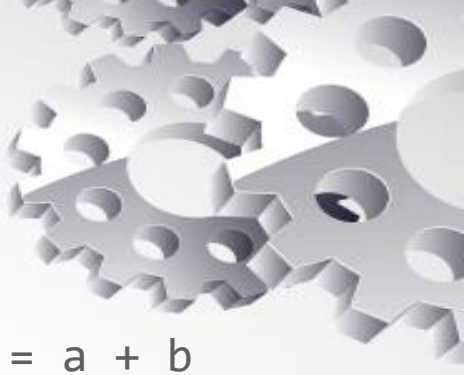
```
let func1 = () => {
    a = 1
    b = 2
    (a + b).toString()
};
func1()

let func2 = () => {
    a = 1
    b = 2
    c = a + b
    (a + b).toString()
};
func2()

// What is the result of calling func1() and func2()
```

What is the result?

Lab 6b: Wrong omission problem



Answer

This is the opposite of the earlier problem. In this case, semi-colon was not inserted to the end of `c = a + b` because the compiler interprets the line `(a + b).toString()` as a continuation of the earlier line. Effectively, the code became:

```
let fun2 = () => {
    a = 1
    b = 2
    c = a + b(a + b).toString()
}
```

The compiler then interprets `b` as a function of `a + b`, hence the error `b is not a function` is thrown.

Lab 6b: Wrong omission problem

Always terminate expression with ;

You should always terminate expressions explicitly using semi-colon ; especially for functions.

```
let func2 = () => {
    a = 1;
    b = 2;
    c = a + b;
    (a + b).toString();
};
func2()
```