Part 1 – Variable & Reassignment Drills

Exercise 1 – Simple `let` change

Write a short program that:

1. Creates a variable named a using `let` and gives it the value 5.
2. Changes the value of a to 8.
3. Prints the value of a using `console.log`.

Before running:

- Write down what you think will be printed.

Exercise 2 – `let` with expression

Write a program that:

1. Creates a `let` variable named b with the value 12.
2. Changes b so that it becomes `b + 3`.
3. Prints b.

Predict the printed value before running.

Exercise 3 – Simple `const`

Write a program that:

1. Creates a `const` variable named `c` with the value `4`.
2. Prints `c`.

Predict:

- What will be printed?

Exercise 4 – `const` reassignment

Write a program that:

1. Creates a `const` variable named `d` with the value `10`.
2. Tries to change `d` to `20`.
3. Tries to print `d`.

Before running, answer:

- Do you expect a value to be printed or an error?
- If there is an error, on which action (reassign or print) do you think it happens?

Exercise 5 – `var` reassignment

Write a program that:

1. Creates a variable named `e` using `var` with the value `2`.
2. Changes `e` to `7`.
3. Prints `e`.

Predict what will be printed.

Exercise 6 – `var` redeclaration

Write a program that:

1. Creates a `var` variable `f` with the value `100`.
2. Declares `f` again using `var`, this time with the value `200`.
3. Prints `f`.

Predict the final printed value of `f`.

Exercise 7 – `let` with multiplication

Write a program that:

1. Creates a `let` variable g with the value 3.
2. Changes g so it becomes `g * 2`.
3. Prints g.

Predict the value that will be printed.

Exercise 8 – `let` redeclare

Write a program that:

1. Creates a `let` variable named h with the value 15.
2. On the next line, tries to create h again using `let`, this time with the value 20.
3. Adds a `console.log` that tries to print h.

Before running:

- Do you expect an error or a printed value?
- If you think there's an error, does the program reach the print line?

Exercise 9 – `const` in an expression (no reassignment)

Write a program that:

1. Creates a `const` variable named `i` with the value 9.
2. Prints the result of `i` plus 1 (without changing `i` itself).

Predict what will be printed.

Exercise 10 – `const` plus equals

Write a program that:

1. Creates a `const` variable named `j` with the value 5.
2. Tries to change `j` so that it becomes `j + 5`.
3. Tries to print `j`.

Predict:

- Do you get a value or an error?
- Does the print line run?

Exercise 11 – Two `let` variables

Write a program that:

1. Creates a `let` variable x with the value 2.
2. Creates another `let` variable y with the value 3.
3. Changes x so that it becomes `x + y`.
4. Prints x.

Predict the printed value of x.

Exercise 12 – Mixing `const` and `let`

Write a program that:

1. Creates a `const` variable `a` with the value `6`.
2. Creates a `let` variable `b` with the value `2`.
3. Changes `b` so that it becomes `a + b`.
4. Prints `b`.

Predict the value printed for `b`.

Exercise 13 – Subtraction with `let`

Write a program that:

1. Creates a `let` variable `m` with value `10`.
2. Changes `m` so that it becomes `m - 4`.
3. Prints `m`.

Predict the output.

Exercise 14 – `var` and multiplication

Write a program that:

1. Creates a `var` variable `r` with value `3`.
2. Changes `r` so that it becomes `r * 5`.
3. Prints `r`.

Predict what will be printed.

Exercise 15 – Several changes with `let`

Write a program that:

1. Creates a `let` variable q with value 4.
2. Changes q so that it becomes `q + q`.
3. Changes q again so that it becomes `q * 2`.
4. Prints q.

Predict the final value of q before running.

Exercise 16 – `const` in a multiplication

Write a program that:

1. Creates a `const` variable k with value 7.
2. Prints the result of `k * 2`.

Predict what will be printed.
(Notice you are *not* changing k, only using it in a calculation.)

Exercise 17 – Another illegal `const` reassignment

Write a program that:

1. Creates a `const` variable z with value 1.
2. Tries to change z so that it becomes `z + 5`.
3. Tries to print z.

Predict:

- Will the program print a value?
- Or will it stop with an error?

Exercise 18 – Two prints with `let`

Write a program that:

1. Creates a `let` variable `t` with value `8`.
2. Changes `t` to `8 + 2`.
3. Prints `t`.
4. Changes `t` again so that it becomes `t + 10`.
5. Prints `t` again.

Before running, write down the **two values** you expect to see, in order.

Exercise 19 – `var` redeclare with expression

Write a program that:

1. Creates a `var` variable `w` with value `5`.
2. Declares `w` again using `var`, but this time gives it the value `w + 5`.
3. Prints `w`.

Predict the printed value.
(Think: what value is used on the right side when you re-declare?)

Exercise 20 – Division and addition with `let`

Write a program that:

1. Creates a `let` variable p with value `10`.
2. Changes p so that it becomes `p / 2`.
3. Changes p again so that it becomes `p + 1`.
4. Prints p.

Predict the final printed value of p.

operators - fill the output:

value 1	value 2	operator	output (students fill)
5	2	+	___
10	3	-	___
4	6	*	___
20	5	/	___
15	4	%	___
2	3	**	___
9	2	+	___
12	7	-	___
6	6	*	___
30	10	/	___
"5"	2	+	___
"5"	2	-	___

5	"2"	*	___
"10"	"3"	+	___
"10"	"3"	*	___
true	5	+	___
false	7	+	___
null	4	+	___
undefined	3	+	___
"hello"	5	-	___

comparison:

value 1	value 2	operator	output (students fill)
5	10	<	___
5	10	>	___
5	5	==	___
5	5	===	___
7	3	>=	___
3	7	<=	___
8	2	!=	___
8	2	!==	___
12	12	===	___
20	10	>	___
"5"	5	==	___
"5"	5	===	___

"10"	"2"	>	___
"abc"	"abd"	<	___
true	false	==	___
true	1	==	___
false	0	===	___
null	undefined	==	___
null	0	==	___
undefined	""	==	___

logical:

value 1	**value 2**	**operator**	**output (students fill)**
true	true	&&	___
true	false	&&	___
true	false	||	___

false	false	||	___
true	true	||	___
1	0	&&	___
1	0	||	___
10	5	&&	___
10	5	||	___
0	5	&&	___
"hello"	"world"	&&	___
"hello"	"world"	||	___
""	"text"	&&	___
""	"text"	||	___
null	"value"	||	___
null	"value"	&&	___
undefined	10	||	___

undefined	10	&&	___
false	"yes"	||	___
"test"	false	&&	___