

## Exercise A: Two Number Adder

Create a JavaScript program that adds two numbers entered by the user.

### Part A

Create a version that works only with JavaScript dialog boxes.

The program should:

- ask the user for two numbers,
- calculate their sum,
- display the result.

### Part B

Create a second version with a simple web interface.

The page should contain:

- two fields where the user can enter numbers,
- a button to perform the addition,
- an area where the result is displayed.

Use a JavaScript function to perform the calculation when the button is clicked.

### Goal

At the end, both versions should calculate the same result, but they should use two different ways of interacting with the user.

## Exercise B: Two Number Calculator

Create a simple calculator with two number inputs and four buttons.

Each button must call a function like `add()`, `multiply()`, ...

The function must calculate the result and display it in an `<output>` element.

**Better version:** Each button must call the same function `calculate(op)`.

Pass the mathematical operation (+, -, \*, /) as a parameter to the function.

## Exercise C: Traffic Light

Create three buttons:

- Green
- Yellow
- Red

All three buttons must call the **same JavaScript function**.

The selected color should be passed to the function as a parameter.

The function then displays a message in an `<output>` element.

Example:

```
<button onclick="trafficLight('green')">Green</button>
<button onclick="trafficLight('yellow')">Yellow</button>
<button onclick="trafficLight('red')">Red</button>
```

Possible messages:

- Green → You may go.
- Yellow → Be careful.
- Red → Stop.

**Extension 1:** Add a traffic light simulator that changes the color of one light according to the button pressed.

**Extension 2:** Add a traffic light simulator that changes that switches the lights on/off according to the button pressed.

Hint: Use ***backgroundColor*** color style property to make these changes.

## Exercise D: Mood Card

Create four buttons:

```
<button onclick="changeMood('happy')">Happy</button>
<button onclick="changeMood('calm')">Calm</button>
<button onclick="changeMood('danger')">Danger</button>
<button onclick="changeMood('night')">Night</button>
```

All buttons call the **same function** and pass a parameter.

Create a box:

```
<div id="box">
  Choose a mood!
</div>
```

The function should change several properties of the box.

For example:

| Mood          | Background | Text             | Other change    |
|---------------|------------|------------------|-----------------|
| <b>Happy</b>  | yellow     | Have a nice day! | large text      |
| <b>Calm</b>   | lightblue  | Relax.           | rounded border  |
| <b>Danger</b> | red        | Warning!         | white bold text |
| <b>Night</b>  | black      | Good night.      | white text      |

## Exercise E: Temperature Advisor

Create a small webpage where the user enters a temperature in °C.

When the user clicks a button, the program should:

- read the temperature from an input field,
- decide which temperature category it belongs to,
- display a suitable message,
- change the color and style of a result box.

Possible categories:

- below 0°C → freezing
- 0°C to 10°C → cold
- 11°C to 20°C → mild
- 21°C to 30°C → warm
- above 30°C → hot

For example:

```
Temperature: [ 25 ]
```

```
[ Check ]
```

```
It is warm.
```

The result box should also visually change.

For example:

| Temperature     | Message     | Suggested style        |
|-----------------|-------------|------------------------|
| <b>below 0</b>  | Freezing!   | light blue background  |
| <b>0–10</b>     | It is cold. | blue background        |
| <b>11–20</b>    | It is mild. | light green background |
| <b>21–30</b>    | It is warm. | orange background      |
| <b>above 30</b> | It is hot!  | red background         |

## Extension 1

Also convert the temperature from Celsius to Fahrenheit.

Formula:

$$^{\circ}\text{F} = ^{\circ}\text{C} \times 1.8 + 32$$

Display both values:

$$25^{\circ}\text{C} = 77^{\circ}\text{F}$$

```
It is warm.
```

## Extension 2

Change more style properties depending on the temperature:

- background color
- text color
- font size
- border
- border radius

## Extension 3

Use a <div> to simulate an analog temperature scale.



## Exercise F: Mars Rover Mission

You control a rover exploring Mars. Choose a distance and terrain, then use the buttons to explore, scan, or recharge.

### Rules

- The rover starts with **100 battery units**.
- Exploring uses battery per kilometre:
  - Flat: **2 units**
  - Rocky: **4 units**
  - Ice: **3 units**
- A scan uses **10 battery units** and adds **1 discovery**.
- Recharging adds **30 battery units**, up to a maximum of 100.
- If there isn't enough battery, the action should not happen.
- Show a warning when the battery is below 20.

### Your task

Start by using Starter HTML

**Then** create a script and make each button work.

Your program should:

1. **Read data** from the distance input and terrain menu.
2. **Calculate** battery use and update the rover's stats.
3. Use **if statements** to check whether the rover has enough battery.
4. **Update the UI** after every action, including a useful message.
5. Make **Recharge** stop at 100 battery units.
6. Make **Reset mission** restore all starting values.

Use one function for each button.



## Mars Rover Mission

Distance to travel (km):

Terrain: Flat

Explore

Scan

Recharge

Reset mission

---

 Battery: 100

 Distance travelled: 0 km

 Discoveries: 0