Unit 8 · Making Web Apps

ANSWER KEY · FOR TEACHERS

Full marks 33

Give the marks for the points in the marking scheme; an answer in other words that says the same is right.

1 Key words

1. JavaScript
2. DOM
3. event
4. listener
5. array
6. NaN
7. specification
8. dark pattern

Each carries 0.5 mark. **K · 4**

2 Label the diagram

1. head
2. title Namaste
3. body
4. button id=”greet” Say namaste

Each carries 0.5 mark. **U · 2**

3 Multiple choice

i. **b.** So that the elements the script looks for are already on the page when it runs. **U · 1**

The browser reads the HTML from the top, and runs a script as soon as it reaches it. A script at the top that looks for `#greet` would find nothing, because the button hasn't been read yet. (There are other ways to wait for the page, but the end of the body is the simplest.)

ii. **b.** No element on the page has `id="total"`: the id is missing, or spelled differently. **U · 1**

`querySelector` gives `null` when nothing matches. Check the HTML: perhaps the element says `id="Total"` (ids care about capital letters) or `id="totals"`, or has no id at all. And # means an id; a class would be `.total`.

iii. **b.** each time the button is clicked, however many times that is **U · 1**

`addEventListener` doesn't run `save`: it hands it to the browser to run whenever the event happens, every time. Writing `save()` with brackets would be a bug: it would run save once, at once, and hand the browser its answer instead of the function.

iv. **b.** a mark of exactly 100 **A · 1**

100 is the largest mark that makes sense: at the edge. 63 is ordinary; “sixty” and −5 are wrong inputs the app should refuse with a message.

v. **b.** honesty: it is a dark pattern **U · 1**

Making one choice easy and the other hard to find pushes people into a choice they might not make. That is a dark pattern: it works for the company, not for the user.

4 What does it print?

MARKING SCHEME **U · 3**

- ✓ `360`, then `3120`: `*` turns the text `'3'` into a number, but `+` with a string joins them as text.
- ✓ `123`: `Number('3')` is 3, and 3 + 120 = 123.
- ✓ `120 NaN`: an empty string becomes 0, and `'abc'` is not a number.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

5 Trace the program

ANSWER **A · 4**

`item`	`bill`	**Output**
(before)	0	
momo	120	
chiya	145	
momo	265	Over 250 at momo
lassi	325	Over 250 at lassi
		Bill: Rs 325

MARKING SCHEME

- ✓ `bill` 120, 145: each round adds the item's price from the object.
- ✓ `bill` 265 and 325.
- ✓ `Over 250 at momo` and `Over 250 at lassi`, in the rounds where the bill is over 250 (not for chiya).
- ✓ `Bill: Rs 325`, once, after the loop.

6 Find and fix the bugs

MARKING SCHEME **A · 3**

- ✓ Line 7: `price` without brackets, so the browser is handed the function to run on each click.
- ✓ Once joined to the string `'Rs '`, `+ 30` joins as text instead of adding.
- ✓ The sum in brackets, `'Rs ' + (plates * 120 + 30)`, with `Number()` on the box's value.

7 Write JavaScript

ANSWER **A · 4**

```
const marks = [72, 65, 80];
let total = 0;
for (const m of marks) {
  total = total + m;
}
if (total > 200) {
  console.log('Well done');
} else {
  console.log('Keep going');
}
```

`marks` never gets a new array, so it is `const`; `total` changes, so it is `let`. The total is 217, so it prints **Well done**. (A counting loop, `for (let i = 0; i < marks.length; i++)` with `marks[i]`, is right too.)

MARKING SCHEME

- ✓ The array and `total` declared, with `const` or `let` (`total` with `let`).
- ✓ The loop: `for (const m of marks) { … }` (or a counting loop), braces round its body.
- ✓ `if (total > 200) { … } else { … }`: the condition in round brackets, braces round each part.
- ✓ `console.log(…)` in place of `print`, with semicolons.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

8 Long answer

ANSWER **A · 4**

The app is right for ordinary whole numbers (24 kg), but not yet correct overall: it shows a long string of digits (2.916666666666667) instead of rounding its answer to a sensible 2 places, and it doesn't check its inputs, so an empty box gives a believable but wrong 0, and a negative weight gives a negative answer instead of a message. Next the team should: round the answer it shows (for example `toFixed(2)`); check for an empty box, NaN and numbers of 0 or below before converting, with a message that says what to type; run all four tests again, plus new edge tests (0 kg, a very large weight); and then test with real users, such as shopkeepers, to see whether it does its job: is it quicker than what they do now, and do they need the answer in kg and dharni both ways?

MARKING SCHEME

- ✓ Right for an ordinary whole number (24 kg), but not yet correct: 7 kg shows a long string of digits instead of 2.92.
- ✓ The inputs aren't checked: an empty box gives a believable but wrong 0, and −5 kg gives a negative answer instead of a message.
- ✓ Next: round what it shows (`toFixed(2)`), and check for an empty box, NaN and 0 or less first, with a message that says what to type.
- ✓ Then run every test again, add edge tests (0 kg, a very large weight), and test with real users such as shopkeepers.

9 Short answer

MARKING SCHEME **U · 2**

- ✓ `innerHTML` reads typed text as HTML, so a script hidden in it would run in the browsers of everyone who sees it (cross-site scripting); `textContent` shows it as plain text.
- ✓ The page's JavaScript runs on the user's own device, where anyone can read, change or skip it; only a check on the server can be trusted.

10 Think

ANSWER **HA · 2**

No. It is a dark pattern: a cost added without asking, hidden in small, pale text, that most customers won't notice. A good cause doesn't make tricking people honest, and customers who find it later stop trusting the shop. Good design would ask openly, with a clear, unticked choice ("Add Rs 10 for the temple?") next to the total, in text as big as the rest; it also fails "for everyone", since small gray text is hard to read for older people and on cheap phones.

MARKING SCHEME

- ✓ Not good design: it is a dark pattern (honesty), a hidden, already-ticked cost most people won't notice, whatever the cause.
- ✓ A better design: ask openly, unticked and clearly shown next to the total (and readable for everyone).

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

11 Activity: Test the converter

(not marked)

WHAT STUDENTS SHOULD FIND

100 gives Rs 160 (ordinary, a pass). 33 should give Rs 52.80 but shows 52.800000000000004, and 0.1 should give Rs 0.16 but shows 0.16000000000000003: 1.6 and 0.1 can't be stored exactly in binary, so the answer must be rounded with `toFixed(2)`. The empty box should give a message but shows Rs 0, because `Number('')` is 0, and −50 should give a message but shows Rs −80 Nepali. Students who fixed the program in the activity should see every row pass; good tests of their own include a very large amount or a negative one.

Marks by level	Knowledge	Understanding	Application	Higher ability	**Total**
Marks	4	11	16	2	**33**

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking