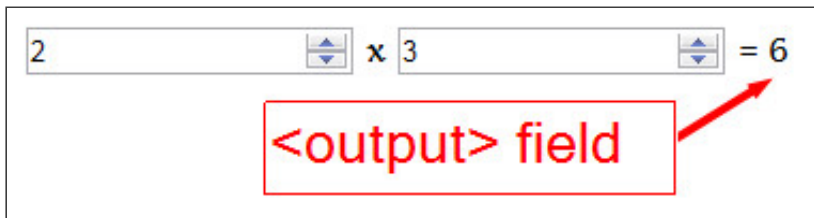# The `<output>` element

The output element represents the result of a computation or a user action. You can see it as a "specialized `<div>` or`<span>`" for displaying interactive results.

TYPICAL USE / INTERACTIVE EXAMPLES

Do not hesitate to play with the source code of these examples online at JS Bin.

Example 1

```
<form oninput="o.value=a.value*b.value">
    <input type="number"name="a" id="a" value="2"> x
    <input type="number"name="b" id="b" value="3"> =
    <output for="a b" name="o">6</output>
</form>
```

The `oninput` event handler directly uses the `<output>` element using the value of its name attribute.

Result (do change the input field values):

2   x 3   = 6

Explanation about the attributes specific to the `<output>` element:

- `for`: a space-separated list containing the elements' ids whose values went into the calculation.

- `name`: the name of the element.

- `form`: associates the `<output>` element with its form owner. The value must be the `id` of a form in the same document. This allows you to place an `<output>` element outside of the `<form>` with which it is associated.

Example 2

Source code:

```
<form oninput="x.value=a.valueAsNumber+b.valueAsNumber">
    <input type="range" name="a" id="a" value="50"/>100
    + <input type="number"name="b" id="b" value="50" />
    = <output name="x" for="a b"></output>
</form>
```

HTML5 introduced new input field properties: `valueAsNumber` and `valueAsDate`. The last example is similar to the previous one except that we use an addition instead of a multiplication.

As input field values are considered as strings by JavaScript, using `x.value = a.value + b.value` would result in a string concatenation instead of a multiplication.

This is why we used the `valueAsNumber` property also introduced by HTML5 for some input fields such as `<input type="range">` and `<input type="number">`, we also encountered the `valueAsDate` properties when we studied `<input`

```
type="date">.
```