

Lesson

Thursday

Introduction to Programming

(/introduction-to-programming)

/ JavaScript and Web Browsers

(/introduction-to-programming/javascript-and-web-browsers)

/ Form Input Types



Text

Cheat sheet

Tips

- Marking your HTML input field as *number*, *date*, or *tel*, doesn't mean it will automatically be a JavaScript **number** type. `type="number"` just means that the form field will only accept number values. But when you access the `.value` property to get the input value, it will still come in as a JavaScript *string*, not a *number*.

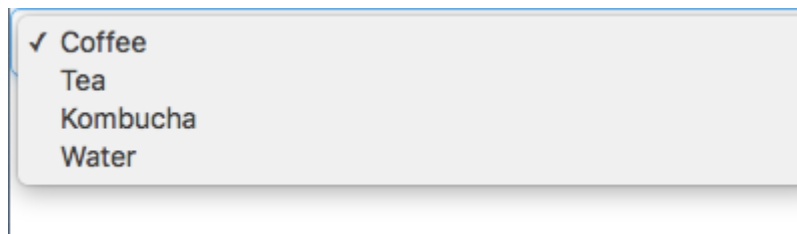
Resources

-  MDN Reference Page on the HTML `<input>` element (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#input_types)
-  An Article on Using HTML5 Input Types (https://developer.mozilla.org/en-US/docs/Learn/Forms/HTML5_input_types)

-  **Bootstrap's Styling Documentation for Forms**
(<https://getbootstrap.com/docs/5.2/components/forms/>)

Code Examples

Select Boxes



HTML

```
<form id="select-form">
  <select id="beverage">
    <option>Coffee</option>
    <option>Tea</option>
    <option>Kombucha</option>
    <option value="4">Water</option>
  </select>
  <button type="submit">Submit Selection</button>
</form>
```

- We can optionally set a `value` attribute to give our `<option>` a different value than the text. For example, the option for "Water" will now have a value of the string "4" .

JavaScript

```
function handleSelect(event) {  
  event.preventDefault();  
  const selection = document.getElementById("beverage").value;  
}  
  
window.addEventListener("load", function() {  
  document.getElementById("select-form").addEventListener("submit", handleSelect);  
});
```

Radio Buttons

- ☒ Chocolate
- ☐ Vanilla
- ☐ Cookies & Cream

HTML

```
<form id="radio-form">
  <label>
    <input type="radio" name="flavor" value="chocolate" checked>
    Chocolate
  </label><br />
  <label>
    <input type="radio" name="flavor" value="vanilla">
    Vanilla
  </label><br />
  <label>
    <input type="radio" name="flavor" value="cookies and cream">
    Cookies & Cream
  </label><br />
  <button type="submit">Submit Selection</button>
</form>
```

Or, not nesting the `<input>` elements and appropriately using `for` and `id` attributes:

```
<form id="radio-form">
  <input type="radio" name="flavor" value="chocolate" id="choc" checked>
  <label for="choc">Chocolate</label><br />
  <input type="radio" name="flavor" value="vanilla" id="van">
  <label for="van">Vanilla</label><br />
  <input type="radio" name="flavor" value="cookies and cream" id="cookies">
  <label for="cookies">Cookies & Cream </label><br />
  <button type="submit">Submit Selection</button>
</form>
```

- `checked` is used to check a radio button by default.
- Including the same `name` attribute and `value` on every radio button input is crucial to the function of this form type. The

`name` attribute groups all of the radio buttons together in order to ensure that only one radio button is selected at a time.

JavaScript

```
function handleRadio(event) {  
  event.preventDefault();  
  const radioSelection = document.querySelector("input[name  
='flavor']:checked").value;  
}  
  
window.addEventListener("load", function() {  
  document.getElementById("radio-form").addEventListener("s  
ubmit", handleRadio);  
});
```

"input[name='flavor']:checked" is a newer, more complicated query selector. Let's break this down:

- `input` targets any inputs in the DOM
- `[name='flavor']` tells the `querySelector` method to look only at elements with a `name` attribute set to `'flavor'`. This ensures that we're looking at all of our radio button inputs, which all share the same `name` attribute and value.
- `:checked` makes sure that we grab the value of only the radio input that is checked.

Date Picker

Date of birth:

mm/dd/yyyy

May 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

HTML

```
<label for="born">Date of birth:</label>
<input id="born" type="date">
```

JavaScript

```
const dob = document.getElementById("born").value;
```

Color Picker

HTML

```
<label for="color">What is your favorite color?</label>
<input id="color" type="color">
```

JavaScript

```
const favoriteColor = document.getElementById("color").value;
```

Button Types — A Review

The `type` attribute that we set for a `<button>` element will change how we can use it in our HTML.

```
<button type="submit">Submit Form</button>
```

A button with a `type` attribute set to `"submit"` is meant to be used in form elements and causes a submission event on a form. The default reaction to a submission event is to refresh the page.

```
<button type="button">Show More Information</button>
```

A button with a `type` attribute set to `"button"` makes the button element have no default behavior and do nothing when pressed by default. If we are targeting a submit event with an event listener and we use `type="button"` on our form's button, our code will break. This is because we can target a click event on a button with `type="button"`, but not a form submission event, so we cannot use `type="button"` in a button within an HTML form. Using `type="button"` on a button element is great for showing/hiding elements, or changing styles.

```
<button>Click Me</button>
```

What about a `<button>` element with no `type` attribute specified? This comes down to context. When unspecified, the default `type` attribute value for buttons is `type="submit"`. So, if the button is used in a form without a `type` attribute, we can still target and respond to a submission event. If a button is used outside of a form, we can also target and respond to a click event.

[Previous \(/introduction-to-programming/javascript-and-web-browsers/understanding-web-apis-event-handling\)](#)

[Next \(/introduction-to-programming/javascript-and-web-browsers/calculator-with-forms-and-branching\)](#)

Lesson 68 of 75

Last updated March 24, 2023

[disable dark mode](#)



© 2023 Epicodus (<http://www.epicodus.com/>), Inc.