EECS 368
Programming Language Paradigms

David O. Johnson

Fall 2022

Reminders

- Assignment 4 due: 11:59 PM, Monday, October 17
- Assignment 5 due: 11:59 PM, Monday, October 31

Any Questions?

In-Class Problem Solution

- 17-(10-3) In-Class Problem Solution.pptx

Any Questions?

Chapter 18 - HTTP and Forms

- ~~The protocol~~
- ~~Browsers and HTTP~~
- ~~Fetch~~
- ~~HTTP sandboxing~~
- ~~Appreciating HTTP~~
- ~~Security and HTTPS~~
- ~~Form fields~~
- ~~Focus~~
- ~~Disabled fields~~
- The form as a whole
- Text fields
- Checkboxes and radio buttons
- Select fields
- File fields
- Storing data client-side

The Form as a Whole

- When a field is contained in a <form> element, its DOM element will have a form property linking back to the form's DOM element.

```
<form action="example/submit.html">
  Name: <input type="text" name="name"><br>
  Password: <input type="password" name="password"><br>
  <button type="submit">Log in</button>
</form>
<script>
  let form = document.querySelector("form");
  console.log(form.elements[1].type);
  // → password
  console.log(form.elements.password.type);
  // → password
  console.log(form.elements.name.form == form);
  // → true
</script>
```

Name:

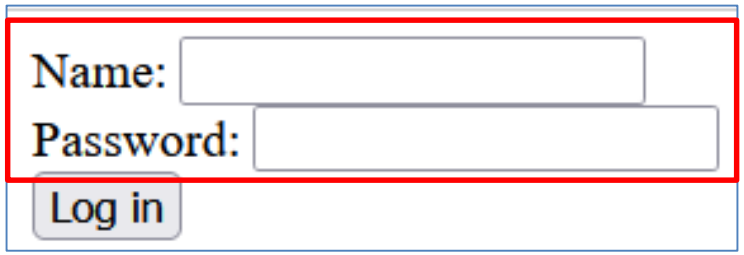
Password:

Log in

The Form as a Whole

- The <form> element, in turn, has a property called elements that contains an array-like collection of the fields inside it.
- The name attribute of a form field determines the way its value will be identified when the form is submitted.
- It can also be used as a property name when accessing the form's elements property, …
- which acts both as an array-like object (accessible by number) and …
- a map (accessible by name).

```
<form action="example/submit.html">
  Name: <input type="text" name="name"><br>
  Password: <input type="password" name="password"><br>
  <button type="submit">Log in</button>
</form>
<script>
  let form = document.querySelector("form");
  console.log(form.elements[1].type);
  // → password
  console.log(form.elements.password.type);
  // → password
  console.log(form.elements.name.form == form);
  // → true
</script>
```



The Form as a Whole

- A button with a type attribute of submit will, when pressed, cause the form to be submitted.
- Pressing enter when a form field is focused has the same effect.
- Submitting a form normally means that the browser navigates to the page indicated by the form's action attribute, using either a GET or a POST request.

```
<form action="example/submit.html">
  Name: <input type="text" name="name"><br>
  Password: <input type="password" name="password"><br>
  <button type="submit">Log in</button>
</form>
<script>
  let form = document.querySelector("form");
  console.log(form.elements[1].type);
  // → password
  console.log(form.elements.password.type);
  // → password
  console.log(form.elements.name.form == form);
  // → true
</script>
```

The Form as a Whole

- But before that happens, a "submit" event is fired.
- You can handle this event with JavaScript and prevent this default behavior by calling preventDefault on the event object.

```
<form action="example/submit.html">
  Value: <input type="text" name="value">
  <button type="submit">Save</button>
</form>
<script>
  let form = document.querySelector("form");
  form.addEventListener("submit", event => {
  console.log("Saving value", form.elements.value.value);
  event.preventDefault();
  });
</script>
```

Value: help Save

Console output:
Saving value help

- Intercepting "submit" events in JavaScript has various uses.
- We can write code to verify that the values the user entered make sense and immediately show an error message instead of submitting the form.
- Or we can disable the regular way of submitting the form entirely, as in the example, and have our program handle the input, possibly using fetch to send it to a server without reloading the page.

Any Questions?

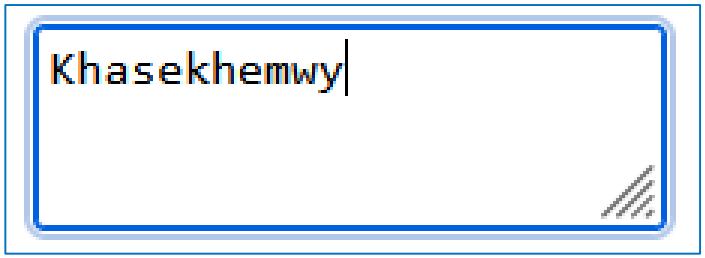
Text Fields

- Fields created by <textarea> tags, or <input> tags with a type of text or password, share a common interface.
- Their DOM elements have a value property that holds their current content as a string value.
- Setting this property to another string changes the field's content.

- The selectionStart and selectionEnd properties of text fields give us information about the cursor and selection in the text.
- When nothing is selected, these two properties hold the same number, indicating the position of the cursor.
- For example, 0 indicates the start of the text, and 10 indicates the cursor is after the 10th character.
- When part of the field is selected, the two properties will differ, giving us the start and end of the selected text.
- Like value, these properties may also be written to.

Text Fields

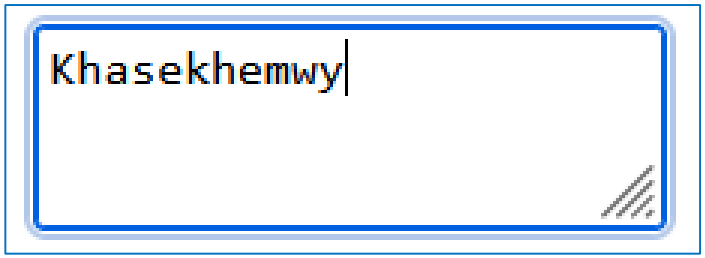
- Imagine you are writing an article about Khasekhemwy but have some trouble spelling his name.
- The following code wires up a <textarea> tag with an event handler that, …
- when you press F2, inserts the string “Khasekhemwy” for you.




```
<textarea></textarea>
<script>
 let textarea = document.querySelector("textarea");
 textarea.addEventListener("keydown", event => {
  // The key code for F2 happens to be 113
  if (event.keyCode == 113) {
   replaceSelection(textarea, "Khasekhemwy");
   event.preventDefault();
  }
 });
 function replaceSelection(field, word) {
  let from = field.selectionStart, to = field.selectionEnd;
  field.value = field.value.slice(0, from) + word +
                field.value.slice(to);
  // Put the cursor after the word
  field.selectionStart = from + word.length;
  field.selectionEnd = from + word.length;
 }
</script>
```

Text Fields

- The replaceSelection function replaces the currently selected part of a text field's content with the given word and …
- then moves the cursor after that word so that the user can continue typing.




```
<textarea></textarea>
<script>
 let textarea = document.querySelector("textarea");
 textarea.addEventListener("keydown", event => {
  // The key code for F2 happens to be 113
  if (event.keyCode == 113) {
   replaceSelection(textarea, "Khasekhemwy");
   event.preventDefault();
  }
 });
 function replaceSelection(field, word) {
  let from = field.selectionStart, to = field.selectionEnd;
  field.value = field.value.slice(0, from) + word +
                field.value.slice(to);
  // Put the cursor after the word
  field.selectionStart = from + word.length;
  field.selectionEnd = from + word.length;
 }
</script>
```

Any Questions?

Checkboxes

- A checkbox field is a binary toggle.
- Its value can be extracted or changed through its checked property, which holds a Boolean value.

```
<label>
 <input type="checkbox" id="purple"> Make this page purple
</label>
<script>
 let checkbox = document.querySelector("#purple");
 checkbox.addEventListener("change", () => {
  document.body.style.background =
  checkbox.checked ? "mediumpurple" : "";
 });
</script>
```

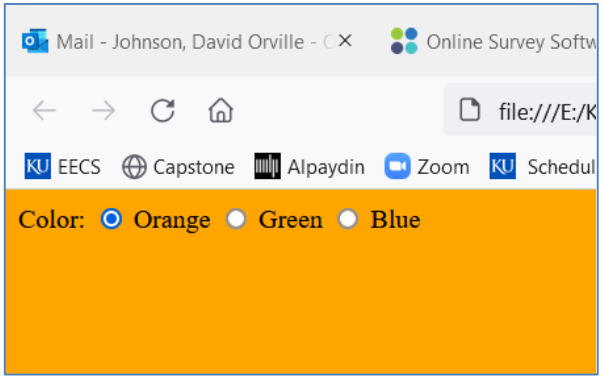
☐ Make this page purple

- The <label> tag associates a piece of document with an input field.
- Clicking anywhere on the label will activate the field, …
- which focuses it and toggles its value when it is a checkbox.

Radio Buttons

- A radio button is similar to a checkbox, but it's implicitly linked to other radio buttons with the same name attribute so that only one of them can be active at any time.

```
Color:
<label>
  <input type="radio" name="color" value="orange"> Orange
</label>
<label>
  <input type="radio" name="color" value="lightgreen"> Green
</label>
<label>
  <input type="radio" name="color" value="lightblue"> Blue
</label>
<script>
  let buttons = document.querySelectorAll("[name=color]");
  for (let button of Array.from(buttons)) {
    button.addEventListener("change", () => {
      document.body.style.background = button.value;
    });
  }
</script>
```

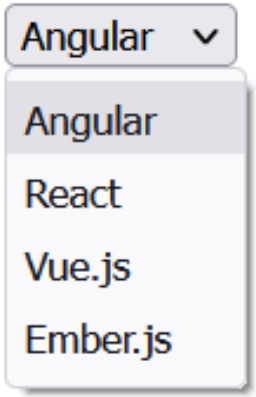


- The square brackets in the CSS query given to querySelectorAll are used to match attributes.
- It selects elements whose name attribute is "color".

Select Fields

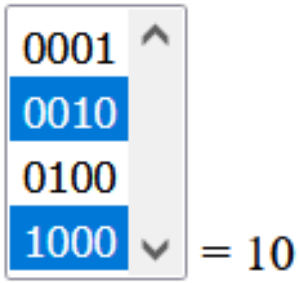
- Select fields are conceptually similar to radio buttons.
- They also allow the user to choose from a set of options.
- But where a radio button puts the layout of the options under our control, …
- the appearance of a <select> tag is determined by the browser.

```
<select id="framework">
   <option value="1">Angular</option>
   <option value="2">React</option>
   <option value="3">Vue.js</option>
   <option value="4">Ember.js</option>
</select>
```



Select Multiple Fields

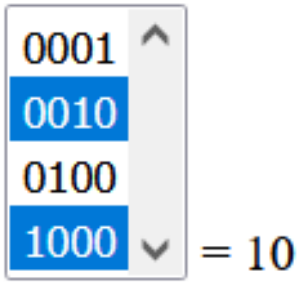
- Select fields also have a variant that is more akin to a list of checkboxes, rather than radio boxes.
- When given the multiple attribute, a <select> tag will allow the user to select any number of options, …
- rather than just a single option.
- This will, in most browsers, show up differently than a normal select field, which is typically drawn as a drop-down control that shows the options only when you open it.




```
<select multiple>
  <option value="1">0001</option>
  <option value="2">0010</option>
  <option value="4">0100</option>
  <option value="8">1000</option>
</select> = <span id="output">0</span>
<script>
  let select = document.querySelector("select");
  let output = document.querySelector("#output");
  select.addEventListener("change", () => {
    let number = 0;
    for (let option of Array.from(select.options)) {
      if (option.selected) {
        number += Number(option.value);
      }
    }
    output.textContent = number;
  });
</script>
```

Select Multiple Fields

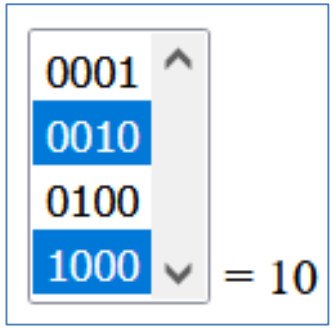
- Each <option> tag has a value.
- This value can be defined with a value attribute.
- When that is not given, the text inside the option will count as its value.
- The value property of a <select> element reflects the currently selected option.
- For a multiple field, though, this property doesn't mean much …
- since it will give the value of only one of the currently selected options.




```
<select multiple>
  <option value="1">0001</option>
  <option value="2">0010</option>
  <option value="4">0100</option>
  <option value="8">1000</option>
</select> = <span id="output">0</span>
<script>
  let select = document.querySelector("select");
  let output = document.querySelector("#output");
  select.addEventListener("change", () => {
    let number = 0;
    for (let option of Array.from(select.options)) {
      if (option.selected) {
        number += Number(option.value);
      }
    }
    output.textContent = number;
  });
</script>
```

Select Multiple Fields

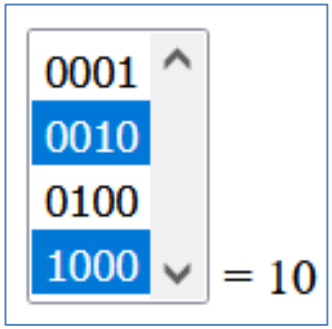
- The <option> tags for a <select> field can be accessed as an array-like object through the field's options property.
- Each option has a property called selected, which indicates whether that option is currently selected.
- The property can also be written to select or deselect an option.




```
<select multiple>
  <option value="1">0001</option>
  <option value="2">0010</option>
  <option value="4">0100</option>
  <option value="8">1000</option>
</select> = <span id="output">0</span>
<script>
  let select = document.querySelector("select");
  let output = document.querySelector("#output");
  select.addEventListener("change", () => {
    let number = 0;
    for (let option of Array.from(select.options)) {
      if (option.selected) {
        number += Number(option.value);
      }
    }
    output.textContent = number;
  });
</script>
```

Select Multiple Fields

- This example extracts the selected values from a multiple select field and …
- uses them to compose a binary number from individual bits.
- Hold control (or command on a Mac) to select multiple options.




```
<select multiple>
  <option value="1">0001</option>
  <option value="2">0010</option>
  <option value="4">0100</option>
  <option value="8">1000</option>
</select> = <span id="output">0</span>
<script>
  let select = document.querySelector("select");
  let output = document.querySelector("#output");
  select.addEventListener("change", () => {
    let number = 0;
    for (let option of Array.from(select.options)) {
      if (option.selected) {
        number += Number(option.value);
      }
    }
    output.textContent = number;
  });
</script>
```

Any Questions?

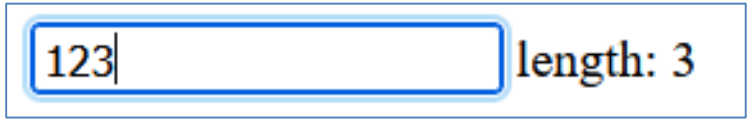
Text Fields

- The "change" event for a text field does not fire every time something is typed.
- Rather, it fires when the field loses focus after its content was changed.

- To respond immediately to changes in a text field, …
- you should register a handler for the "input" event instead, …
- which fires for every time the user types a character, …
- deletes text, or …
- otherwise manipulates the field's content.

- The following example shows a text field and a counter displaying the current length of the text in the field:

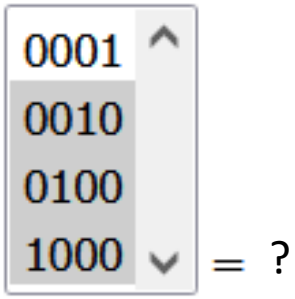
```
<input type="text"> length: <span id="length">0</span>
<script>
 let text = document.querySelector("input");
 let output = document.querySelector("#length");
 text.addEventListener("input", () => {
  output.textContent = text.value.length;
 });
</script>
```



Any Questions?

In-Class Problem

1. What value will this code show for the question mark (?)?
2. Comment each line of JavaScript showing how it calculated your answer for No. 1.




```
<select multiple>
  <option value="1">0001</option>
  <option value="2">0010</option>
  <option value="4">0100</option>
  <option value="8">1000</option>
</select> = <span id="output">0</span>
<script>
  let select = document.querySelector("select");
  let output = document.querySelector("#output");
  select.addEventListener("change", () => {
    let number = 0;
    for (let option of Array.from(select.options)) {
      if (option.selected) {
        number += Number(option.value);
      }
    }
    output.textContent = number;
  });
</script>
```