LC 101

Unit 3 - JavaScript

March 9, 2017

Input Fields

- As we learned in Unit 2, HTML has many types of input fields.

```
<p><input type="text" value="abc"> (text)</p>
<p><input type="password" value="abc"> (password)</p>
<p><input type="checkbox" checked> (checkbox)</p>
<p><input type="radio" value="A" name="choice">
   <input type="radio" value="B" name="choice" checked>
   <input type="radio" value="C" name="choice"> (radio)</p>
<p><input type="file"> (file)</p>
```

Input Fields

```
<textarea>
This is
a multi-line
text field.
</textarea>

<select>
  <option>Pancakes</option>
  <option>Pudding</option>
  <option>Ice cream</option>
</select>
```

Focus

- Unlike most other HTML elements, input fields can have keyboard focus
  - Become the active element and main receiver of keyboard input
- Getting focus generates a **`focus`** event
- Losing focus generates a **`blur`** event
- We can cause an element to get or lose focus with the **`focus()`** and **`blur()`** methods

```
document.getElementById("someField").focus();
```

Focus

- In HTML, we can use the **`autofocus`** attribute to indicate an element should have the focus by default

```
<input type="text" autofocus>
```

- The **`tabindex`** attribute can be used to give the order in which to move the focus when the user hits the tab key
    - Can be used to skip focusable fields

```
<input type="text" tabindex=1>
<a href=".">(help)</a>
<button onclick="console.log('ok')" tabindex=2>OK</button>
```

Disabled

- Fields can be disabled in HTML via the **`disabled`** attribute

```
<button disabled>Can’t touch this</button>
```

- Can also disable/enable fields via the disabled property on DOM nodes

```
document.getElementById("field1").disabled = true;
document.getElementById("field2").disabled = false;
```

Forms

- Input elements can exist on their own or as part of a form
  - If input elements are part of a form then we can access them via the `elements` property of the form node, either by index or by name

```
<form id="loginForm" action="login" method="post">
  Name: <input type="text" name="name"><br>
  Password: <input type="password" name="password"><br>
  <button type="submit">Log in</button>
</form>

var form = document.getElementById("loginForm");
console.log(form.elements[1].type);  // outputs "password"
console.log(form.elements.password.type)  // outputs "password"
```

Form Submit Event

- We can do something before form submission by catching the **`submit`** event
- We can prevent the default submit action by calling the **`preventDefault()`** method
    - Why would we do this?
        - We can validate the form before making the request to the server
        - We can send the data using **`XMLHttpRequest`** instead of a normal request to avoid loading a new page (more on this in a later class)

```
var form = document.getElementById("loginForm");
form.addEventListener("submit", function(event) {
  // …
  event.preventDefault();
});
```

Form Submit Event

- Old style of attaching a submit handler and suppressing the default action
  - Set the `onsubmit` property of the form to an anonymous function
  - Return false to suppress the default form submission action
    - Return true to allow the submit to continue

```
var form = document.getElementById("loginForm");
form.onsubmit = function() {
  // …
  return false;
};
```

Local Storage

- A web page can store a limited amount of data locally in the browser
    - Typically limited to a few MB total
    - Stored as key-value string pairs
- Like cookies, a web page can only access its own data
- Unlike cookies, the data is not sent to the server

```
localStorage.setItem("aKey", "someValue");
var value = localStorage.getItem("aKey"));
localStorage.removeItem("aKey");
```

Session Storage

- Session storage is similar to local storage but is automatically cleared when the tab or window is closed

```
sessionStorage.setItem("aKey", "someValue");
var value = sessionStorage.getItem("aKey"));
sessionStorage.removeItem("aKey");
```