DLP Term-Time Projects: HTML, CSS, and JavaScript Workshop

http://dlp.io, 10/20/15

Neel Mehta (neelmehta@college.harvard.edu)

Sample code from the Digital Literacy Project's (http://dlp.io) first Term-Time Project comp meeting, in which we learned: - HTML - Styling with Bootstrap - JavaScript - jQuery - ES6

Try all these code snippets and more at http://is.gd/dlpcomp.

HTML

Examples

```
<!-- basic elements -->
<h1>This is DLP.</h1>

<p>We teach CS to middle school students.</p>

<button>Self destruct</button>

<!-- nested elements -->
<p>
  Feeling lucky? If so <button>Click me</button>.
</p>

<!-- attributes -->
<input type="text" placeholder="Type something...">
<button>Search</button>

<!-- ul and multi-nested elements -->
<p>Some cities:</p>
<ul>
  <li>Boston</li>
  <li>Cambridge</li>
  <li>Somerville</li>
</ul>

<!-- links -->
<p>Copyright 2015 <a href="http://dlp.io">the Digital Literacy Project</a>.</p>
```

Challenge

Write HTML to generate a page like this:

Challenge solution

```
<h1>Favorite distractions</h1>
<ul>
  <li><a href="https://twitter.com">Twitter</a></li>
  <li><a href="https://netflix.com">Netflix</a></li>
</ul>

<p>
  What's your favorite distraction?
  <input type="text" placeholder="Website name">
  <input type="text" placeholder="Website URL">
  <button>Submit</button>
</p>
```

Bootstrap

Examples

```
<p>
  Just another paragraph.
</p>

<!-- contextual classes -->
<p class="lead">
  Big fancy text™.
</p>

<!-- multi-classes -->
<p class="lead pull-right">
  They pulled me to the right!
</p>

<!-- buttons and color classes-->
<p>
  <button>Yawn</button>
  <button class="btn btn-primary">Submit</button>
  <button class="btn btn-danger">Self destruct</button>
  <button class="btn btn-default btn-lg">Mega</button>
</p>

<!-- styling other elements -->
<p>
  <a href="http://getbootstrap.com" class="btn btn-success">Get Bootstrap</a>
</p>

<!-- list groups -->
<p>Some cities:</p>
<ul class="list-group">
  <li class="list-group-item">Boston</li>
  <li class="list-group-item list-group-item-success">Cambridge</li>
  <li class="list-group-item">Somerville</li>
</ul>

<!-- inputs -->
<input type="text" placeholder="Fancy input" class="form-control">
<button class="btn btn-default">Search</button>

<!-- grid system -->
<div class="row">
  <div class="col-sm-6">
    left half
  </div>
  <div class="col-sm-6">
    right half
  </div>
</div>

<!-- advanced grid system -->
<div class="row">
  <div class="col-sm-4">
    a third
  </div>
  <div class="col-sm-2">
    a sixth
  </div>
  <div class="col-sm-6">
    the rest
  </div>
</div>
```

Challenge

This is the HTML we generated for the previous challenge:

```
<h1>Favorite distractions</h1>
<ul>
  <li><a href="https://twitter.com">Twitter</a></li>
  <li><a href="https://netflix.com">Netflix</a></li>
</ul>

<p>
  What's your favorite distraction?
  <input type="text" placeholder="Website name">
  <input type="text" placeholder="Website URL">
  <button>Submit</button>
</p>
```

Now add Bootstrap styling to this code so that the page looks like this:

Challenge solution

```
<h1>Favorite distractions</h1>

<div class="row">
    <div class="col-sm-8">
        <ul class="list-group">
          <li class="list-group-item"><a href="https://twitter.com">Twitter</a></li>
          <li class="list-group-item"><a href="https://netflix.com">Netflix</a></li>
        </ul>
    </div>
    <div class="col-sm-4">
        <p>
          What's your favorite distraction?
          <input type="text" placeholder="Website name" class="form-control">
          <input type="text" placeholder="Website URL" class="form-control">
          <button class="btn btn-primary btn-lg">Submit</button>
        </p>
    </div>
</div>
```

Basic JavaScript

HTML

```
<p class="lead" id="output"></p>
```

JavaScript

```
// utility printing function (don't worry about this)
let write = (text) => {
  let message = $("<p>").html(text + "");
  $("#output").append(message);
};

// variables and functions
let number = 5;
number = number * 3;
write(number);

// creating functions
let writeExcited = (text) => {
    write(text + "!!!")
};

// calling functions
writeExcited("hey");

// calling multi-argument functions
let bigger = Math.max(2, 7);
write(bigger);
```

jQuery

Example

HTML

```
<div class="row">
  <div class="col-sm-8">
    <p class="lead">
      The counter is <strong id="counter">0</strong>.
    </p>
  </div>
  <div class="col-sm-4">
    <button class="btn btn-success btn-lg" id="add">
      +
    </button>
    <button class="btn btn-danger btn-lg" id="subtract">
      -
    </button>
    <button class="btn btn-info btn-lg" id="clear">
      0
    </button>
  </div>
</div>
```

JavaScript (first run)

```
let count = 0;

// event handlers, $, html
$("#add").on("click", () => {
  count = count + 1;
  $("#counter").html(count);
});

// subtract: do this yourself!
$("#subtract").on("click", () => {
  count = count - 1;
  $("#counter").html(count);
});

// clear; do this yourself!
$("#clear").on("click", () => {
  count = 0;
  $("#counter").html(count);
});
```

New JavaScript (using functions)

```
let count = 0;

let update = (newCount) => {
  count = newCount;
  $("#counter").html(newCount);
};

$("#add").on("click", () => {
  update(count + 1);
});

// subtract: do this yourself!
$("#subtract").on("click", () => {
  update(count - 1);
});

// clear: do this yourself!
$("#clear").on("click", () => {
  update(0);
});
```

New JavaScript (additional error checking)

```
let count = 0;

let update = (newCount) => {
  // ensure that the count never goes below 0
  if (newCount >= 0) {
    count = newCount;
    $("#counter").html(newCount);
  }
};

$("#add").on("click", () => {
  update(count + 1);
});

$("#subtract").on("click", () => {
  update(count - 1);
});

$("#clear").on("click", () => {
  update(0);
});
```

Advanced JavaScript and ES6

HTML for all examples in this section

```
<p class="lead" id="output"></p>
```

Arrays

```
// utility printing function
let write = (text) => {
  let message = $("<p>").html(text + "");
  $("#output").append(message);
};

// working with arrays
let numbers = [1,2,3];
write(numbers);
write(numbers.length);

// reading and writing at indices
write(numbers[0]);
numbers[0] = numbers[2];
write(numbers[0]);

// mixed types in arrays
let stuff = ["Toothpaste", 29, 1.55];
write(stuff);

// Challenge: swap elements 0 and 2
let temp = stuff[0];
stuff[0] = stuff[2];
stuff[2] = temp;
write(stuff);
```

Map

```
// utility printing function
let write = (text) => {
  let message = $("<p>").html(text + "");
  $("#output").append(message);
};

let numbers = [1,2,3];
write(numbers);

// mapping one array to another with a function
let squared = numbers.map((x) => {
  return x * x;
});
write(squared);

// function shorthand
let squared2 = numbers.map(x => x * x);
write(squared2);

// Challenge: write "11,12,13"
let tenAdded = numbers.map(x => x + 10);
write(tenAdded);
```

Combining maps

```
// utility printing function
let write = (text) => {
  let message = $("<p>").html(text + "");
  $("#output").append(message);
};

let numbers = [1,2,3];

// defining functions
let square = (x) => x * x;
let plusOne = (x) => x + 1;

// mapping with variables
let squared = numbers.map(square);
write(squared);
let added = numbers.map(plusOne);
write(added);

// composing maps
let squaredPlusOne = numbers.map(square).map(plusOne);
write(squaredPlusOne);

// Challenge: use square and plusOne to write "9,16,25"
let challenge = numbers.map(plusOne).map(plusOne).map(square);
write(challenge);
```

Objects

```
// utility printing function
let write = (text) => {
  let message = $("<p>").html(text + "");
  $("#output").append(message);
};

// object syntax
let mySchool = {
  name: "Harvard",
  year: 1636
};

let theirSchool = {
  name: "Yale",
  year: 1701
};

// reading from objects
write(mySchool.year);
write(theirSchool.name);

// writing to objects
mySchool.year = theirSchool.year;
write(mySchool.year);
```

Arrays of objects

```
// utility printing function
let write = (text) => {
  let message = $("<p>").html(text + "");
  $("#output").append(message);
};

// arrays of objects
let schools = [
  { name: "Harvard", year: 1636 },
  { name: "Yale", year: 1701 },
  { name: "Princeton", year: 1746 }
];

// mapping over object arrays
let years = schools.map(school => school.year);
write(years);
```