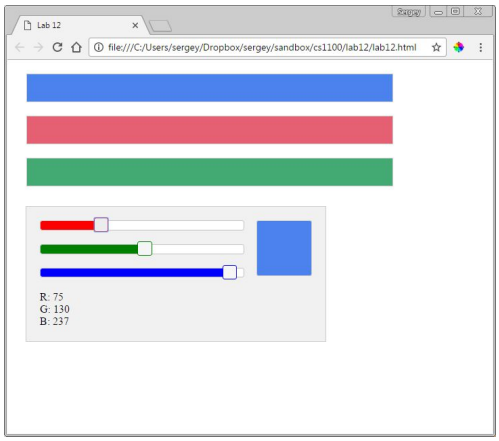
7CS 1100: Web Development: Client-Side Coding / Fall 2016
Lab 12: Exploring jQuery UI

Description

The goal of this lab is to try out the jQuery UI API. You will use the jQuery UI slider control (or widget)., You will build a simple color picker (similar to the one on the jqueryuni.com site, but sufficiently different). You'll will start with a simple slider, then you'll add a few more elements and explore some of the functionality and options provided to you by the jQueryUI API. As a result, you will have a color picker that looks like this:



At the end of the lab, you will be able to add your color picker tool to any other web page (that you have downloaded) and adjust the color of most elements on that page, like this:

Through this exercise, you will see how jQuertyUI components can interact with your own code (HTML, CSS and JavaScript).

Step 0

Download the lab12.zip file and unzip it. Before you start, take a look at the code in lab12.html and lab12.css.

HTML

- You have several divs that you will use for the various elements:
    - colorpicker: this is a "wrapper" and its purpose is layout only
    - red, green, blue: these are the actual sliders for 3 colorsswatch: the div to demo the currently selected color
    - values: numeric values of red, green and blue
    - sample1, 2, 3: three divs that we will use as targets for our new colors
- A reference to the jqueryui css file
- A reference to your own css file
- A reference to the jquery library
- A reference to the jqueryui library
- A reference to your own js file

CSS

Most of these styles will make sense once we implement the slider functionality. None of them are essential: all are for aesthetic purposes only (except the swatch and the samples: without a specified width and height those divs would have been invisible on the page - but our sliders would still work).

Step 1

First, take a look at the slider demo pages: http://jqueryui.com/slider/

Look at the examples (use the menu on the right). Make sure to click the view source link at the bottom of the examples (you only need to click it once).

Now look at the slider API: you will find it by clicking the API Documentation in the main menu, then Widgets, then Slider. You do not need to read all the documentation, but do read the intro paragraphs and then take a quick look at the options, methods and events. This is where you will look when you need to figure out how to use a method (or to check what you can do with this widget). Again, API is your friend: it tells you what you can do with a code library.

Step 2

Create your lab12.js file. Let's start building!

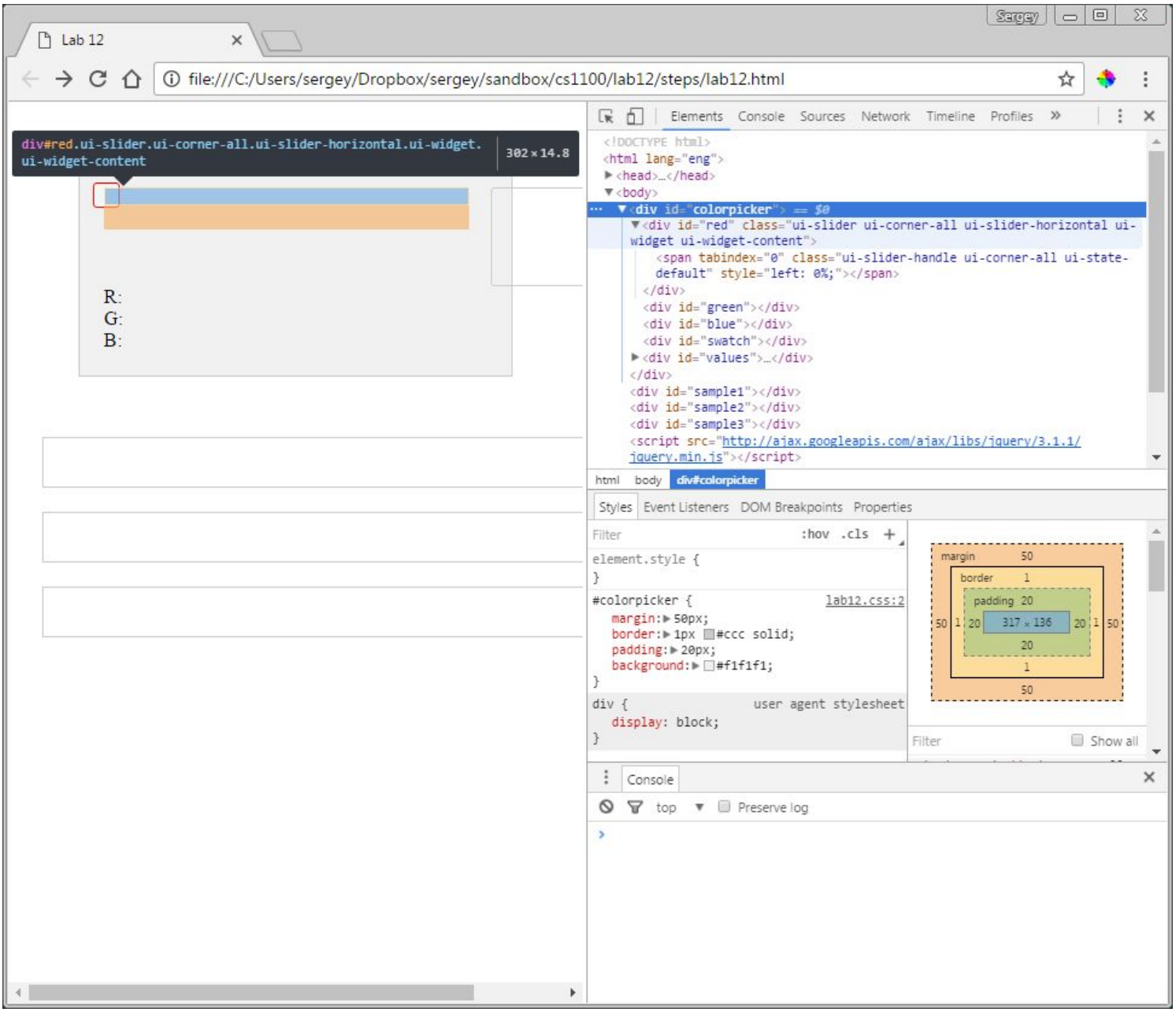
First, add the jQuery "wrapper":

```
$(function() {
    //all of your code goes here
});
```

Now let's make a slider:

```
$("#red").slider();
```

This code creates a jQuery object that wraps the HTML element that we've obtained using the CSS selector "#red". Then it calls the slider() method, which transforms that element into a slider by adding some more HTML and assigning to it CSS classes (that are defined in the jqueryui css file). Open the inspect tool (same as console, but choose the elements tab). Compare the red and green/blue divs - you'll see the code that has been added.



Now that you know what exactly is happening, let's create all 3 sliders:

```
$("#red").slider();
$("#green").slider();
$("#blue").slider();
```

Or, we could even better (improve your CSS selector):

```
$("#red, #green, #blue").slider();
```

Step 3

Time to move beyond the default functionality.

The slider's purpose is to select a value by sliding a handle. The default minimum value is 0 (which works for us. We need to specify a maximum value of 255 (because we are working with 8-bit colors which range from 0 to 255). For this, we would set the max option: http://api.jqueryui.com/slider/#option-max

We also need to make our sliders recognize a range of values - for styling only. With a range, we get access to a class name **ui-slider-range** we can style, which represents the selected range that we can then style (see our css file). We will use the rang option: http://api.jqueryui.com/slider/#option-range

We also need to handle an event: when a handle is moved, something must happen. In fact, we need to handle 2 events: slide and change (both happen):
http://api.jqueryui.com/slider/#event-slide
http://api.jqueryui.com/slider/#event-change

We'll handle them using the same function that we'll define later. So, here's your new slider code:

```
$("#red, #green, #blue").slider({
      max: 255,
      range: "min",
      slide: updateColor,
      change: updateColor
});

function updateColor() {
}
```

Pay attention to the syntax: we pass the options using the following notation:

{ property: value, property: value, etc... }

Of course, if the value is a string, you need quotations.

Wer also provide an empty function definition for updateColor() - we'll fill out the details later.

Check your web page - use the slider.

Step 3

Actually, we are almost done! We have three sliders. When we move them, a function gets called. The only thing left is to specify what happens ***inside that function***.

3.1. Get the values of each slider and, for convenience, assign them to variables:

```
var red = $("#red").slider("value");
var blue = $("#blue").slider("value");
var green = $("#green").slider("value");
```

We selected the **slider**, then called its **.slider()** method , requesting the value of its **value** property.

Now let's display these values using our HTML span elements:

```
$("#rval").text(red);
$("#gval").text(green);
$("#bval").text(blue);
```

Finally, let's change the background color of our "swatch" element. How do we do that??

That's easy: you have the values of red, green and blue! We could convert them to hex, but we don't have to: we can use the rgb() css notation instead!

If we have the following values: r=255, g=0, b=0 (that's very bright red), we would need this: "rgb(255, 0, 0)" for our css. So, step 1 is to build up this string:

```
var newColor = "rgb(" + red + ", " + green + ", " + blue + ")";
```

The next step is to assign the new color to the swatch as its new background:

```
$("#swatch").css({backgroundColor: newColor});
```

Try it out!

3.2. When we first load the page, our swatch doesn't have a color, and there are no values indicated for the R,G,B labels. It's super easy to fix that: just call the updateColor() function right ***after*** you initialize the sliders!

```
$("#red, #green, #blue").slider({
      max: 255,
      range: "min",
      slide: updateColor,
      change: updateColor
});

updateColor();
```

Step 3

What's the use of a color picker on a webpage if you can't mess with the page? Let's change that!

We want to be able to do the following:
- select any part of the page
- and then apply our new color to the selected element.

Specifically, we want to **add an event handler to everything** that will handle **an appropriate event** (double-click works best so that we don't select our color picker) by **assigning the clicked element to be the new target** of our color shenanigans.

First, create a variable that will hold the current target. Assign the swatch div - that will be our default.

```
var $target = $("#swatch");
```

Now add our event handler to everything:

```
$("*").on("dblclick", function(e) {
	$target = $(this);
	e.stopPropagation();
	updateColor();
});
```

`e.stopPropagation();` is required to stop the event from propagating (if we don't do this, clicking an element will send the same event to the element's parent: clicking a div would also fire the click event on the body element - so you would be changing the color of the body on every case - you can try this without this line.

`$target = $(this);` changes the current target to whatever we clicked.

`updateColor();` And once we clicked, we might as well change its color right away!

Finally, in our updateColor() function, we need to set the background color of the target element:

```
$target.css({backgroundColor: newColor});
```

And lastly, for convenience, let's make our color picker draggable - so we can move out of the way when selecting other elements:

```
$("#colorpicker").draggable();
```

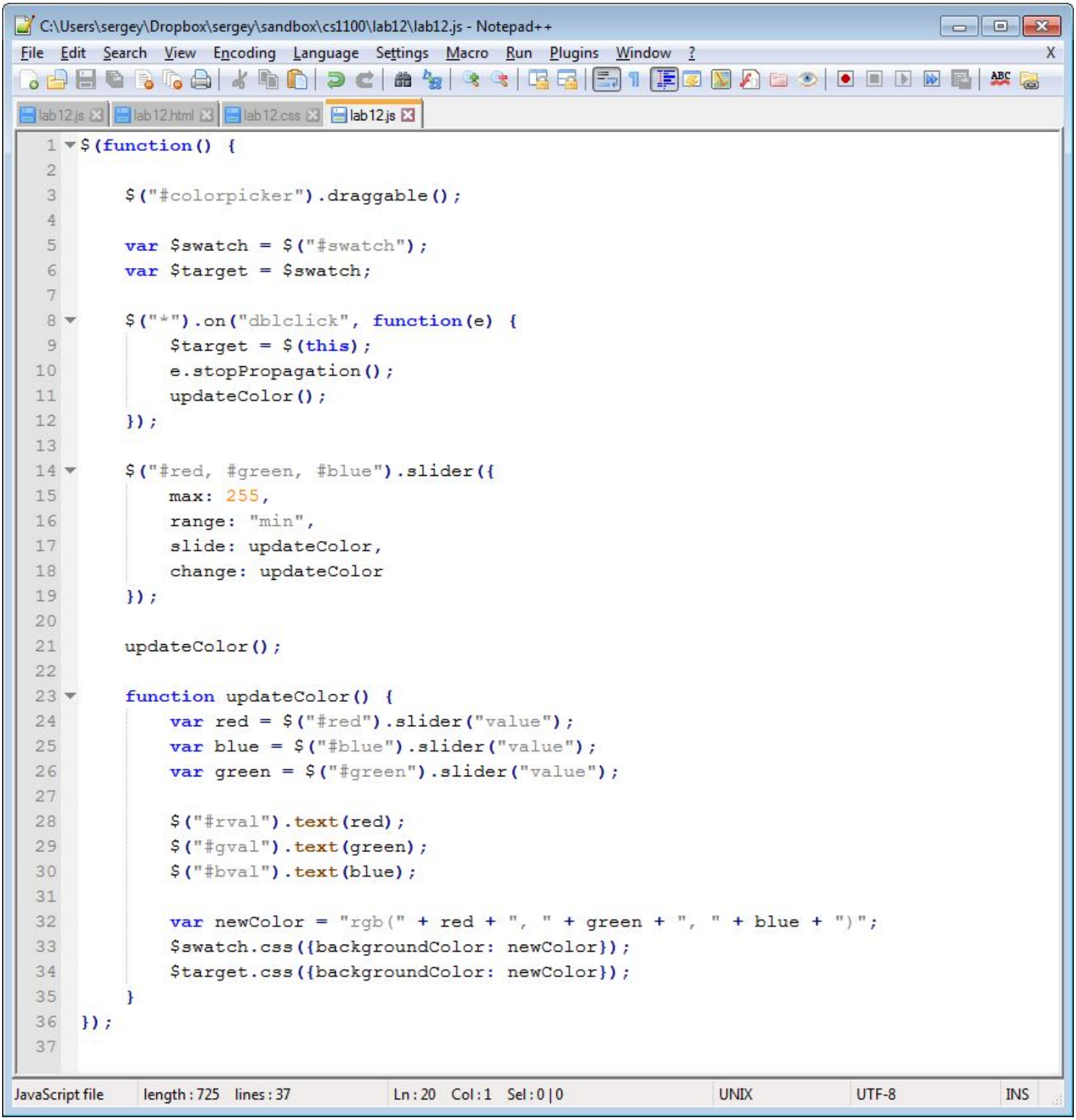
WE can also change the CSS (***in your lab12.css file***) and make its position fixed, so that no matter how far we scroll, our color picker is at the top of the screen:

```
#colorpicker {
	margin: 50px;
	border: 1px #ccc solid;
	padding: 20px;
	background: #f1f1f1;
	position: fixed;
}
```

Run your code. You are done! Congratulations!

For fun, try to apply this to other pages:

1. Save a webpage to your local folder
2. Open the html and add the required references:
    a. A reference to the jqueryui css file
    b. A reference to your own css file
    c. A reference to the jquery library
    d. A reference to the jqueryui library
    e. A reference to your own js file
3. Past the colorpicker div into the html.
4. Have fun!



Submit your work

Submit **lab12.js** to eLearning:
https://bb9.uni.edu > log in with CatID > our course > Course Content > Labs > Lab 12

For Your Reference: This is what your code should look like in the end: