jQuery

Warmup

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>This and event handlers</title>
  <style>
div {
    margin: auto;
    width: 300px; height: 300px;
    text-align: center;
}
.red {
    background-color: red;
}
.blue {
    background-color: blue;
    color: white;
}
  </style>
  <script>
window.onload = init;
function init() {
    var div = document.getElementById("clickme");
    div.onclick = clickHandler;
}
function clickHandler() {
    // finish this
}
  </script>
</head>
<body>
  <div class="red" id="clickme">Click Me!</div>
</body>
</html>
```

Hint:

- div is an object.
- We're assigning a method to a property of that object.
- When that method gets called, how do we refer to the object itself in its method? Like we do in the show method of our dog objects?

```
var rainbow = {
    name: "Rainbow",
    weight: 16,
    breed: "Beagle",
    likesFetching: true,
    bark: function bark() {
        console.log("Woof woof!");
    },
    show: function() {
        console.log(this.name + " is a " +
          this.weight +
          " pound Beagle who likes fetching");
    }
};
```

jsfiddle

Now:

- Add on to the code so that when you click the color of the box toggles between red and blue.

  `jsfiddle`

Compare:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>This and event handlers</title>
  <style>
div {
    margin: auto;
    width: 300px;
    height: 300px;
    text-align: center;
    background-color: red;
}
.blue {
    background-color: blue;
    color: white;
}
  </style>
  <script>
function init() {
    var div = document.getElementById("clickme");
    div.onclick = toggle;
}

function toggle() {
    var currentClass = this.getAttribute("class");
    if (currentClass === "red") {
        this.setAttribute("class", "blue");
    } else {
        this.setAttribute("class", "red");
    }
}
  </script>
</head>
<body>
  <div class="red" id="clickme">Click Me!</div>
</body>
</html>
```

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>This and event handlers</title>
  <script src="http://code.jquery.com/jquery-latest.min.js"></scrip
  <style>
div {
    margin: auto;
    width: 300px;
    height: 300px;
    text-align: center;
    background-color: red;
}
.blue {
    background-color: blue;
    color: white;
}
  </style>
  <script>
$(function() {
    $("#clickme").click(function() {
        $(this).toggleClass("blue");
    });
});
  </script>
</head>
<body>
  <div class="red" id="clickme">Click Me!</div>
</body>
</html>
```

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>This and event handlers</title>
  <script src="http://code.jquery.com/jquery-
latest.min.js"></script>
  <style>
div {
    margin: auto;
    width: 300px;
    height: 300px;
    text-align: center;
    background-color: red;
}
.blue {
    background-color: blue;
    color: white;
}
  </style>
  <script>
$(function() {
    $("#clickme").click(function() {
        $(this).toggleClass("blue");
    });
});
  </script>
</head>
<body>
  <div class="red" id="clickme">Click Me!</div>
</body>
</html>
```

Dissecting $

First: what *is* $?

- Load a web page into the browser that links to jQuery.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>This and event handlers</title>
  <script src="http://code.jquery.com/jquery-latest.min.js"></script>
  <script>
  </script>
</head>
<body>
</body>
</html>
```

- Open the console and type: window.jQuery

- Then type window.$ - for both of these you should see something like this:

```
function (a,b){return new m.fn.init(a,b)}
```

- Link to uncompressed source code.

Dissecting $

$ is a multi-purpose function. It can:

- Call a function when the DOM has finished loading:

```
$(function() { … });
```

- Select elements from the DOM:

```
$("#clickme")…
```

- Create new HTML:

```
$("<div>I'm a new chunk of HTML</div>")…
```

- Wrap a plain DOM object to turn it into a jQuery object:

```
$(this)…
```

- In JavaScript, functions are objects, so you'll sometimes see $ used as an object too:

```
$.each(…)
```

- Link to API documentation

jQuery objects

```
$("#clickme")
```

- Returns a jQuery object: a custom object with jQuery methods and properties. If you save the jQuery object in a variable, a convention is to use a $ in the name, to distinguish it from a plain DOM object:

  ```
  var $div = $("#clickme");
  ```

- Demo $div. [menu] in the console.
- In jQuery, we often "chain" method calls. Most jQuery methods return a jQuery object. So…

  ```
  $("#clickme").toggleClass("blue");

  $("#clickme").toggleClass("blue").fadeOut().fadeIn();
  ```

- Each time we call a method, the jQuery object is returned, so we can call another method on it.

jQuery objects - getting a DOM object back

```
var $div = $("#clickme");
```

- Returns a jQuery object: a custom object with jQuery methods and properties.
- What if you need the plain DOM object?

```
var div = $("#clickme").get(0);
```

- Then of course, you can turn that back into a jQuery object any time:

```
var $anotherDivVariable = $(div);
```

What if multiple objects match?

```
var $allDivs = $("div");
```

- The jQuery object returned from a selection is always an array. If the array has more than one thing in it, then you get an array of length > 1.

- If you call a jQuery method on the result, the method will apply to *all* the matching elements!

```
$("div").fadeOut().fadeIn();

$("div").toggleClass("blue");
```

- If you use array syntax to get an item from the array, you'll get a plain DOM object:

```
var $allDivs = $("div");
var firstDiv = $allDivs[0];
```

- To get one item from the jQuery array, as a jQuery object, use first(), last(), next().

```
var $allDivs = $("div");
var $firstDiv = $allDivs.first();
```

Callbacks (usually, event handlers)

```
$(function() {
    $("#clickme").click(function() {
        $(this).toggleClass("blue");
    });
});
```

- Read this as "When you click on the <div> with the id "clickme", call the function that I'm passing to the click() function."

- Notice: we are passing a function to a function! It's totally fine to do this in JavaScript (JavaScript is kind of unusual in this regard).

- Inside the click handler function, **this** is set to the *DOM object* you clicked on (just like a DOM click handler). If you want to call another jQuery function on that object, you have to wrap it first!

- Passing **this** to the $() function turns it into a jQuery object, so then we can call the toggleClass method on that object (toggleClass is a jQuery function).

Callbacks to do stuff when something else is done.

```
$("#clickme").fadeOut("slow", function() {
        console.log("Fade is done!");
});
```

- Read this as "Fade out the <div> with the id "clickme" slowly, and when it's done fading, display "Fade is done" in the console.

- Each jQuery method is a little different, so use the API Documentation to look up the methods and see what options they have:

  http://api.jquery.com

Project

Todo list manager: Silver

- Re-implement your to do list from yesterday using jQuery

Todo list manager: Platinum

- Form to create a new to do item.
- Add new to do items to a list when you click the button.
- Use <li> class "done" to keep track of if an item is done or not. (Hint: use toggleClass to toggle done items).
- If you mark all the items in a list "done", ask the user if they want to remove all the items in the list.
    - Select items using the "done" class and the "check" class.
    - Use the length of the results to determine if all items are done.
    - Add new items by creating new HTML for each piece and putting it together using the jQuery append() method.