

# **Comp 424 - Client-side Web Design**

---

Fall Semester 2016 - Week 9 Notes

Dr Nick Hayward

# Contents

---

- jQuery
  - *basics*
- HTML5, CSS, & JS - example
  - *more options...*
- jQuery
  - *manipulate the DOM*
- HTML5, CSS, & JS - example
  - *cont'd...*

# jQuery - basics - part I

---

## *intro*

- jQuery offers us a number of useful tools and options for building web apps
- packaged, prepared JavaScript library
  - *a lot easier to work with, and develop for, than standard JavaScript*
- features simpler syntax and a concise set of options for manipulating the DOM
  - *often simply quicker and easier to write our apps with jQuery than JavaScript*
- jQuery is an inherently expressive approach to working with JavaScript
  - *in particular, manipulating the DOM*
- consistent approach to handling events in the DOM
- includes useful, simplified approach to adding AJAX functionality

# jQuery - basics - part 2

---

## selectors

- jQuery works with selectors using a similar concept as CSS
- we can use CSS selectors as a jQuery selector

```
$("div")
$("p")
$(".note-input")
$(".note-input button")
$("p:nth-child(even)")
...
```

- jQuery may share many selectors with CSS
  - some cases where jQuery will slightly differ
- adds useful set of pseudoclasses and pseudoelements not in CSS

```
$("p:parent")
```

- use the above to find all paragraphs with children, including text
- a jQuery extension, and not part of the CSS specification

# jQuery - basics - part 3

---

## *manipulate the DOM*

```
<body>

  <!-- document header -->

  <header>

    <h3></h3>

    <p></p>

  </header>

  <!-- document main -->

  <main>

    <!-- note input -->

    <section class="note-input">

      <h5>add note</h5>

      <input><button></button>

    </section>

    <!-- note output -->

    <section class="note-output">

    </section>

  </main>

  <!-- document footer -->

  <footer>

    <p></p>

  </footer>

</body>
```

- benefits of using jQuery is the ease it offers for manipulating the DOM
- add elements, delete them, move them around...

# jQuery - basics - part 4

---

## add elements

- add a new element to our app
  - *simply append or prepend to a given position in the DOM*

```
//append note text to note-output
$(".note-output").append($note);
```

- adds our new element, and content to the DOM
  - *end of the selected element in document*

```
//append note text to note-output
$(".note-output").prepend($note);
```

- prepend to the document
  - *adds to the end of the selected element*
- additional options in jQuery, such as `prependTo( )`
- differ slightly on the target for the content
- useful to select an element, then add to another elsewhere in DOM

# jQuery - basics - part 5

---

## **remove elements**

- also remove elements from the DOM
- easiest option is to use the `remove ( )` function on a given selector

```
$("p:nth-child(even)").remove();
```

- also empty an element, remove all child elements from selected element
  - *remove all of the notes, those we added in paragraph elements*

```
$(".note-output p").empty();
```

- also temporarily remove elements from the window

```
$note.fadeOut("slow");
```

- elements are not removed from the DOM, their style is updated

```
display: none;
```

# jQuery - basics - part 6

---

## events and async

- jQuery uses a standard pattern for events and handling

```
//handle user event for `add` button click
$(".note-input button").on("click", function(e) {
    ...
});
```

- allows us to set up listeners for many user triggered events
- commonly known as **event-driven** or **asynchronous** programming
- main difference with more traditional procedural patterns, is the way we use **callbacks**
  - *allow us to set functions for later execution*
- functions are set as parameters, then executed at the appropriate, required time
- callbacks are not only appropriate for interaction or user events
- use them throughout our programming to schedule functions and execution

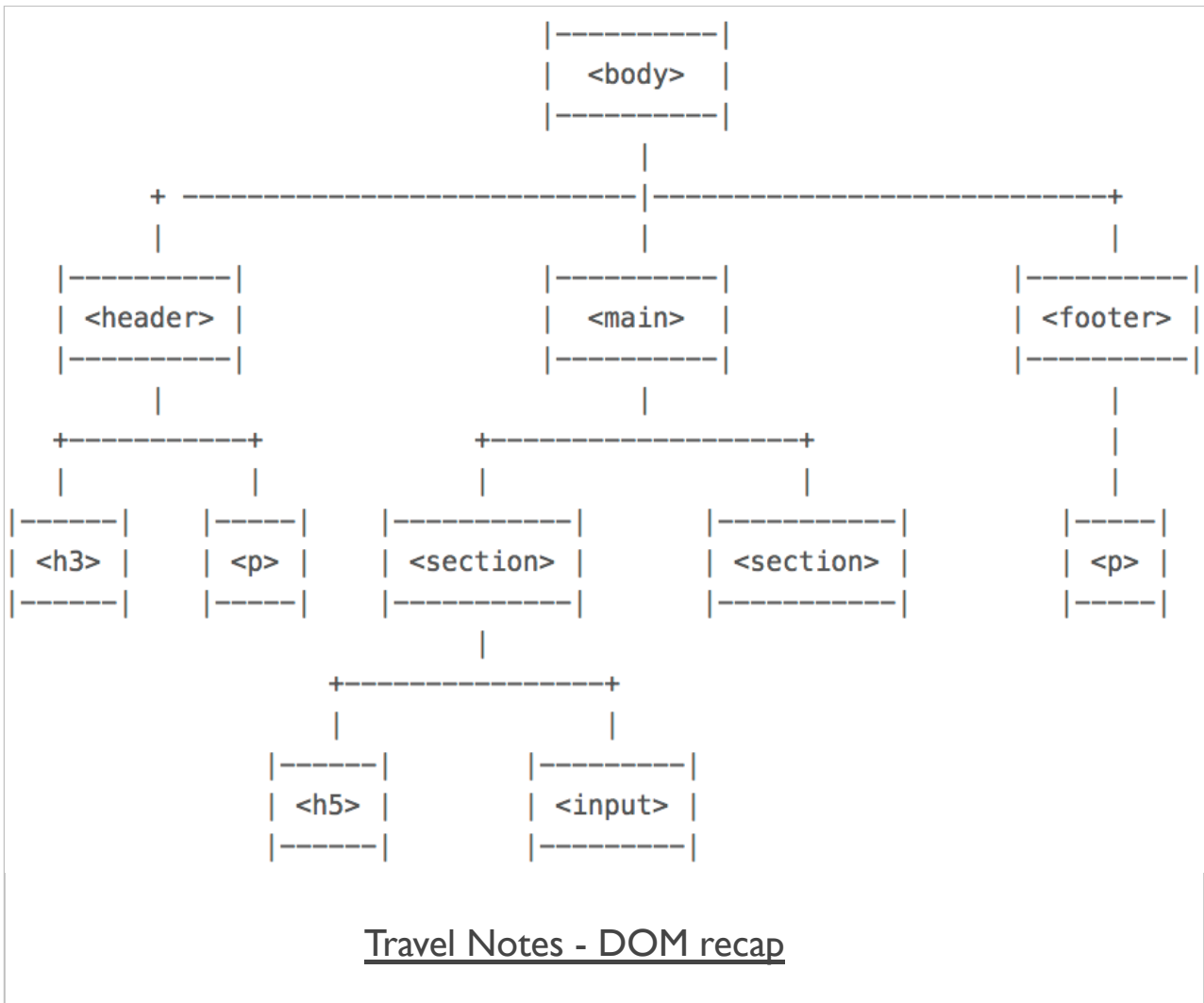
```
setTimeout(function() {
    ...
}, 2000);
```

- an issue with **asynchronous** programming
  - *often simply being aware of the execution order or sequence of events*



# Image - HTML5, CSS, & JS - DOM recap

---



# Image - Travel Notes - Series I - recap

---

## travel notes

*record notes from various cities and places visited...*

### add note

add

have fun in St Tropez

ride the tram in Nice

play golf in Mougins

app's copyright information, additional links...

Travel Notes - Series I - Demo 8 recap

# HTML5, CSS, & JS - example - add-ons

---

*new features and add-ons...*

- delete all notes
- delete a single note
- new event handlers
- additional styling

# HTML5, CSS, & JS - example - part I

---

## *delete option - all notes*

- standard `remove()` function in jQuery

```
$("#p").remove();
```

- option to **delete all** notes from `.note-output`
- add a new toolbar for note controls and options

```
<section class="note-controls">
  <button id="notes-delete">Delete all</button>
</section>
```

- then add some simple styling for this new toolbar

```
/* note controls */
.note-controls {
  margin: 10px 0 10px 0;
  padding: 2px;
  border-bottom: 1px solid #dedede;
  display: none;
}

/* simplify default button styles for note controls */
.note-controls button {
  padding: 2px;
  margin: 2px;
  border-radius: 0;
  border: 1px solid #dedede;
  cursor: pointer;
}
```

# HTML5, CSS, & JS - example - part 2

---

## ***delete option - all notes***

- note controls toolbar is hidden, by default in the CSS
- need some way to check its visibility as we add our notes
  - *no notes, then the toolbar is not required*

```
//check element visibility - expects single element relative to display:none
function checkVisible(element) {
  if (element.is(":hidden")) {
    element.fadeIn();
  }
}
```

- simply checking a passed element to see whether it is hidden
  - *then fadeIn( ) as necessary*
- can update this method later on to check hidden and visible
- call this function as required

```
checkVisible($(".note-controls"));
```

# HTML5, CSS, & JS - example - part 3

---

## *delete option - all notes*

- add a note, the `.note-controls` toolbar is shown
  - ***delete all*** button now becomes available to our users

```
//handle deletion of all notes
$("#notes-delete").on("click", function(e) {
    var $note = $(".note-output p");
    $(this).parent().hide();
    $note.remove();
});
```

- creating a new handler for the click events on the `#notes-delete` button
- hides its own container, the notes toolbar
- then removes all of the notes, `p`, from the `.note-output` section

# HTML5, CSS, & JS - example - part 4

---

## *JS code so far*

```
//check element visibility - expects single element relative to display:none
function checkVisible(element) {
    if (element.is(":hidden")) {
        element.fadeIn();
    }
}
...
//handle deletion of all notes
$("#notes-delete").on("click", function(e) {
    var $note = $(".note-output p");
    $(this).parent().hide();
    $note.remove();
});
```

- DEMO 1 - travel notes - series 2

# HTML5, CSS, & JS - example - part 5

---

*delete option - all notes*

- still making an assumption notes exist in the note-output section
- add an additional function to check element exists in the DOM or not
- use jQuery's `length()` function

```
$("p").length()
```

- new function for checking elements in the DOM is as follows,

```
//check elements exists
function checkExist(element) {
    if (element.length) {
        return true;
    } else {
        return false;
    }
}
```



# HTML5, CSS, & JS - example - part 6

---

## **delete option - all notes**

- updated delete all notes option to include check for notes
- call `checkExist()` function in conditional statement

```
//handle deletion of all notes
$("#notes-delete").on("click", function(e) {
    //set note selector
    var $note = $(".note-output p");
    //check $note exists
    if (checkExist($note) === true) {
        //hide note-controls
        $(this).parent().hide();
        //remove all notes
        $note.remove();
    }
});
```

- DEMO 2 - travel notes - series 2

# Image - Travel Notes - Series 2 - demo 2

---

## travel notes

*record notes from various cities and places visited...*

### add note

add

Delete all

stroll along the Promenade des Anglais in Nice

lose money in Monaco

meet Picasso in Antibes

be seen in Cannes

app's copyright information, additional links...

Travel Notes - Series 2 - Demo 2

# HTML5, CSS, & JS - example - part 7

---

## **delete option - per note**

- consider adding a single delete option per note
- allowing a user to selectively delete their chosen note
  - *regardless of hierarchical position within the `.note-output` section*
- design decisions for such an option might include
  - *do we offer a selection option, such as checkboxes, to select one or more delete items*
  - *perhaps a single delete button per note*
  - *a drag and drop to delete option*
  - *there are many different ways to present and use this option*
- programmatically follow a similar pattern for deletion of the note
- three jQuery functions can help us remove elements from a document
  - `remove()`
  - `detach()`
  - `replaceWith()`

# jQuery - removing elements - quick overview

---

- used `remove ( )` function with delete all notes
  - *best used to remove elements permanently from a document*
  - *will **unbind** any attached event handlers for elements being removed*
  - *will return reference to removed elements, but not the original bound events*
- `detach ( )` often used for any temporary removal requirements
  - *eg: update a lot of the DOM, detach affected elements, then insert later...*
  - *retains its event handlers, and we can add these elements later*

```
$("p").detach();
```

- then append the attached elements as required

```
var $detachP = $("p").detach();  
$detachP.appendTo("#detached");
```

- `replaceWith ( )` replaces an element, or group of elements, with passed element
- event handlers for the replaced elements are unbound

```
var $replacedP = $(".note-output p").first().replaceWith("<p>replaced...</p>");
```

# HTML5, CSS, & JS - example - part 8

---

## **delete option - per note**

- simply need to delete the selected note
  - use the same `remove()` function for single and all notes
- add option per note to allow user to delete a required note
- add a delete button for each note
  - add programmatically with each new note

```
function createButton(buttonClass, buttonText) {  
  var $button = $('<button class="'+buttonClass+'">'+buttonText+'</button>');  
  return $button;  
}
```

- new function allows us to create simple buttons as required
  - a specified class and button text passed as parameters
  - use function to build required delete button in `createNote()` function

```
//create delete button  
var $delete_button = createButton("note-delete", "delete");
```

# HTML5, CSS, & JS - example - part 9

---

*delete option - per note*

- append delete option to note
  - *before adding note to the DOM in createNote function*

```
function createNote() {  
  ...  
  //set content for note  
  $note.html($note_text.val());  
  //append delete button to each note  
  $note.append($delete_button);  
  ...  
}
```

# HTML5, CSS, & JS - example - part 10

---

## ***delete option - per note***

- need to bind a click event to the dynamically created delete note button
- delete button is being added to the DOM dynamically
  - *need to add handler for single note deletion event to existing DOM element*
  - *add handler to parent `.note-output` and then new `button.note-delete`*

```
$(".note-output").on("click", "button.note-delete" , function() {  
    //delete parent note  
    $(this).parent().remove();  
    //set note selector  
    var $note = $(".note-output p");  
    //check for empty notes, and then remove note-controls  
    if (checkExist($note) === false) {  
        //hide note-controls  
        $(".note-controls").hide();  
    }  
});
```

- DEMO 3 - travel notes - series 2

# Image - Travel Notes - Series 2 - demo 3

---

## travel notes

*record notes from various cities and places visited...*

### add note

add

Delete all

breakfast in Antibes delete

lunch in Nice delete

dinner in Monaco delete

app's copyright information, additional links...

Travel Notes - Series 2 - Demo 3



# HTML5, CSS, & JS - example - part II

---

## *delete option - per note*

- now allow our users to delete a single note
- single note option is awkward at the moment
- simply allow a user to either mouseover or select a note to show additional options
  - *showing the available delete button*
- enable a user to select their note of choice
  - *need to bind a click event to a note*

```
//handle click event per note  
$(".note-output").on("click", "p", function() {  
  ...  
}
```

- user selects a note
  - *no check for previous other visible delete buttons*
  - *ensure only delete button for selected note is shown*

# Image - HTML5, CSS, & JS - too many delete buttons

---

## travel notes

*record notes from various cities and places visited...*

### add note

cannes note

nice note

monaco note

antibes note

app's copyright information, additional links...

Travel Notes - Week 6 - Too many delete buttons

# HTML5, CSS, & JS - example - part 12

---

*delete option - per note*

- return to our earlier function, `checkVisible()`
- modify to allow better abstraction and usage
- modify to test for visibility
  - *then simply return a boolean value*

```
//check element visibility - expects single element relative to display:none
function checkVisible(element) {
    //check if element is hidden or not
    if (element.is(":hidden")) {
        return true;
    } else {
        return false;
    }
}
```

- also need to modify check for the `.note-controls` in `createNote()` function

```
...
//check visibility of note controls
if (checkVisible($(".note-controls")) === true) {
    $(".note-controls").fadeIn();
}
...
```

# HTML5, CSS, & JS - example - part 13

---

## ***delete option - per note***

- updated handler for note selection now checks for visible delete buttons

```
//handle click event per note
$(".note-output").on("click", "p", function() {
    //check if other delete buttons visible
    if (checkVisible($(".button.note-delete")) === true) {
        $(".button.note-delete").hide();
    }
    $(this).children(".button.note-delete").show();
});
```

- bind handler for the user clicking on a note
- check whether other delete buttons are visible on any other notes
  - *if visible, we can simply hide these delete buttons*
  - *then show the delete option for the currently selected note*
- later abstract this function to handle other options associated with each note
- DEMO 4 - travel notes - series 2

# HTML5, CSS, & JS - example - part 14

---

## style note(s)

- add some additional styling to our notes
  - *start with some changes to the design of each note*
  - *then considered the overall `.note-output` section*
- remove styling for alternating notes, set uniform style per note

```
/* note paragraph output */  
.note-output p {  
  margin: 10px;  
  padding: 10px;  
  border: 1px solid #b1c4b1;  
  cursor:pointer;  
}
```

- need to add some styling for our delete button, and position it within each note

```
/* note delete button */  
.note-output p button.note-delete {  
  display: block;  
  padding: 5px;  
  margin: 5px 5px 10px 0;  
  border-radius: 0;  
  border: 1px solid #dedede;  
  cursor: pointer;  
}
```

# HTML5, CSS, & JS - example - part 15

---

## style note(s)

- add some styling for the button's hover pseudo-class
  - *acts as useful feedback to the user that the button is an active element*

```
.note-output p button.note-delete:hover {  
  background-color: #aaa;  
  color: #fff;  
}
```

- also consider adding some similar feedback to our note
  - *a sign of active as the user moves their mouse cursor over each note*

```
/* note paragraph output hover */  
.note-output p:hover {  
  border: 1px solid #1a3852;  
}
```

- DEMO 5 - travel notes - series 2

# HTML5, CSS, & JS - example - part 16

---

## style note(s)

- a couple of issues that still need to be fixed in the application
  - *first issue is lack of consistency in styling our buttons*
- fixed by abstracting our CSS styling for a default button
  - *specific button styles can be added later*

```
/* default button style */  
button {  
  padding: 2px;  
  margin: 2px;  
  border-radius: 0;  
  border: 1px solid #dedede;  
  cursor: pointer;  
}
```

- removed the need for a ruleset to style the button for
  - *adding a note, delete all notes, and the single delete button per note*

# HTML5, CSS, & JS - example - part 17

---

## style note(s)

- also create a default ruleset for a button hover pseudo-class
  - *again reducing our need for repetition in the stylesheet*

```
/* default button hover style */  
button:hover {  
  background-color: #aaa;  
  color: #fff;  
}
```

- iterative development is fine
  - *continue to abstract styles, overall design, and logic as we develop an application*



# HTML5, CSS, & JS - example - part 18

---

*style note(s)*

- second issue is the expected interaction with each note
  - *issue is simply that a user cannot choose to remove this option*
- should be able to toggle its view and options
- update interaction by modifying handler for click event on a note
  - **NB:** `toggle()` for events was removed in jQuery 1.9
  - *build our own*

```
//handle click event per note
$(".note-output").on("click", "p", function() {

    //check if other delete buttons visible
    if (checkVisible($(".button.note-delete")) === true) {

        //set all siblings to active=false to ensure checks are correct
        $(this).siblings().attr("active", "false");

        $(".button.note-delete").hide();
    }

    //then handle click event for current note
    if (!$$(this).attr("active") || $(this).attr("active") === "false") {

        $(this).attr("active", "true");

        $(this).children("button.note-delete").show();
    } else if ($(this).attr("active") === "true") {

        $(this).attr("active", "false");

        $(this).children("button.note-delete").hide();
    }

});
```

- DEMO 6 - travel notes - series 2

# HTML5, CSS, & JS - example - part 19

---

*a few extras to consider...*

- alternative layouts
  - *grid*
  - *squares*
  - *snippet view*
  - *table*
  - *lists...*
- notifications
- snippets with expansion
- split views
  - *note snippet with contextual/media per note...*
- drag and drop delete
- filters
- sort options
- tags
- much, much more...

# Image - Square notes - a bit of fun

---

## travel notes

*record notes from various cities and places visited...*

### add note

cannes

nice

monaco

antibes

frejus

st tropez

eze

app's copyright information, additional links...

Travel Notes - Week 6 - Squares

- DEMO - travel notes - squares

# JS Objects - quick recap - part I

---

- important JavaScript primitive
  - *one of the most frequently used as well*
- created with curly braces,

```
var object1 = {  
  "a": "nine",  
  "b": "ten"  
};
```

- access internal variables of this object using the dot . operator

```
console.log(object1.a);
```

- update the value of an internal variable

```
object1.a = "amelia";
```

# JS Objects - quick recap - part 2

---

- also create an empty variable, and then assign values as necessary

```
var object1 = {};
```

- an object can contain variables with values of different types
- store variables in an object with types such as strings, arrays, and even other objects
- function variables behave just like any other variables in JavaScript
  - *we can also store them in our objects as needed*

```
var $a = $("p");  
$a.hide();
```

- simply attach a function to a jQuery object

# JSON - quick recap

---

- a JSON object is effectively a JavaScript object
  - *contained within curly braces*

```
{  
  "country": "France",  
  "city": "Marseille"  
}
```

- objects can contain multiple name/value pairs
- object stored in the form of a string
- to send a JS object
  - *create it in the application's code*
  - *then convert it to a string*
  - *finally use it as required*
- a lot of the AJAX is abstracted to JavaScript libraries

# JSON - pros and cons

---

## **useful pros**

- more concise, less verbose than XML and HTML
  - *potentially faster execution of data...*
- regularly used with JavaScript
  - *includes good support*
- language agnostic, interoperability
  - *can be used with many different programming languages*
- can also be called from many different domains
  - *eg: JSON-P...*

## **some cons**

- may present security risk
  - *malicious content due to JavaScript XSS*
  - *need to verify source for JSON...*
- syntax is precise, unforgiving

# JS and JSON - functions

---

- creating some JSON string is easy enough
- also easily create a JSON string from a JavaScript object
  - *and vice-versa*
- use the JavaScript `stringify` function

```
var jsonObject1 = JSON.stringify(object1);  
console.log(jsonObject1);
```

- similarly parse a JSON string to a JS object

```
var object2 = JSON.parse(jsonObject1);  
console.log(object2);
```



# AJAX and JSON - part I

---

## intro

- AJAX is as a simple way to load data
  - *often new or updated data*
  - *into a current page without having to refresh the browser window*
- common form of data for work with AJAX is JSON
- many common usage scenarios and examples for AJAX
  - *autocomplete in forms*
  - *live filtering of search queries*
  - *real-time updates for content and data streams*
- also use AJAX to help us load data behind the scenes
  - *preparing content for our users before a specific request is received*
  - *helps to speed up page responses and data load times*
- AJAX uses an asynchronous model for processing requests
- user can continue to perform various tasks, queries, and work
  - *whilst the browser itself continues to load data*
- inherent benefit of AJAX should include
  - *a more responsive site, intuitive usage and interface experience*

# AJAX and JSON - part 2

---

## *asynchronous model*

- traditional synchronous model normally stops a page
  - *until it has loaded and processed a requested script*
- AJAX enables a browser to request data from the server
  - *without this synchronous pause in usage*
- AJAX's **asynchronous processing model**
  - *often known as **non-blocking***
  - *allows a page to load data and process user's interactions*
- server responds with the requested data
  - *an event will be fired by the browser*
  - *event can then call a function to process the data*
  - *often JSON, XML, or simply HTML*
- browser will use an **XMLHttpRequest** object to help handle these AJAX requests
- browser will not wait for a response

# Demos

---

## **Travel notes app - series 2**

- DEMO 1 - Travel notes - demo 1
- DEMO 2 - Travel notes - demo 2
- DEMO 3 - Travel notes - demo 3
- DEMO 4 - Travel notes - demo 4
- DEMO 5 - Travel notes - demo 5
- DEMO 6 - Travel notes - demo 6

# References - JS & Libraries

---

- [jQuery](#)
- [jQuery API](#)
- [jQuery :parent selector](#)
- [MDN](#)
  - *[MDN - JS Objects](#)*
- [W3 - JS Object](#)
- [W3 - JS Performance](#)