Comp 388/424 - Client-side Web Design

Fall Semester 2015 - Week 10

Dr Nick Hayward

Contents

- Feedback & grades
- Ajax, JSON & jQuery continued
    - *recap - get the data*
    - *jQuery deferred, promise*
    - *handling errors*

Ajax, JSON & jQuery - part 1

jQuery Deferred

- jQuery provides a useful solution to the escalation of code for asynchronous development
- known as the `$.Deferred` object
  - *effectively acts as a central despatch and scheduler for our events*
- with the **deferred** object created
  - *parts of the code indicate they need to know when an event completes*
  - *whilst other parts of the code signal an event's status*
- **deferred** coordinates different activities
  - *enables us to separate how we trigger and manage events*
  - *from having to deal with their consequences*

Ajax, JSON & jQuery - part 2

using deferred objects

- now update our AJAX request with **deferred** objects
- separate the asynchronous request
  - *into the initiation of the event, the AJAX request*
  - *from having to deal with its consequences, essentially processing the response*
- separation in logic
  - *no longer need a success function acting as a callback parameter to the request itself*
- now rely on `.getJSON()` call returning a **deferred** object
- function returns a restricted form of this **deferred** object
  - *known as a* ***promise***

```
deferredRequest = $.getJSON (
  "file.json",
  {format: "json"}
);
```

Ajax, JSON & jQuery - part 3

using deferred objects

- indicate our interest in knowing when the AJAX request is complete and ready for use

```
deferredRequest.done(function(response) {
  //do something useful...
});
```

- key part of this logic is the `done()` function
- specifying a new function to execute
  - *each and every time the event is successful and returns complete*
  - *our AJAX request in this example*
- **deferred** object is able to handle the abstraction within the logic
- if the event is already complete by the time we register the callback via the `done()` function
  - *our **deferred** object will execute that callback immediately*
- if the event is not complete
  - *it will simply wait until the request is complete*

Ajax, JSON & jQuery - part 4

handling errors with deferred objects

- also signify interest in knowing if the AJAX request fails
- instead of simply calling `done()`, we can use the `fail()` function
- still works with JSONP
  - *the request itself could fail and be the reason for the error or failure*

```
deferredRequest.fail(function() {
  //report and handle the error...
});
```

Ajax, JSON & jQuery - part 5

example

- add the option to read and write from a JSON file
- we'll use AJAX for these requests
- initially we can consider our application as follows
    - *read data from JSON file*
    - *load initial data to application*
    - *save new data to the JSON file*
- no edit features for now
    - *with the exception of deleting notes from JSON file*
- add edit features with DB

Ajax, JSON & jQuery - part 6

example - JSON

- test reading and loading JSON file and data
- ignore standard AJAX pattern
  - *passing two callbacks, success and error*
- use `deferred` and `promise`
- inital JSON for Travel Notes app

```
{
  "travelNotes": [{
    "created": "2015-10-12T00:00:00Z",
    "note": "a note from Cannes..."
  }, {
    "created": "2015-10-13T00:00:00Z",
    "note": "a holiday note from Nice..."
  }, {
    "created": "2015-10-14T00:00:00Z",
    "note": "an autumn note from Antibes..."
  }]
}
```

Ajax, JSON & jQuery - part 7

example - deferred

- start by submitting a query for the required JSON file
- then retain the deferred object we're using for tracking
- then indicate interest in knowing when AJAX request is complete

```
//load main app logic
function loadApp() {
  "use strict";

    var $deferredNotesRequest = $.getJSON (
      "docs/json/notes.json",
      {format: "json"}
    );

    $deferredNotesRequest.done(function(response) {
        console.log("tracking json...");
    });

};
$(document).ready(loadApp);
```

Ajax, JSON & jQuery - part 8

example - deferred

- `done()` method is the key part
- helps us specify the required logic to execute
  - *when the request is complete*
- if the given event has already completed as callback is registered via `done()`
  - *deferred object will execute required callback immediately*
- if not, it will simply wait until request is complete
- respond to an error
  - *add `fail()` method for errors handling and reporting*

Ajax, JSON & jQuery - part 9

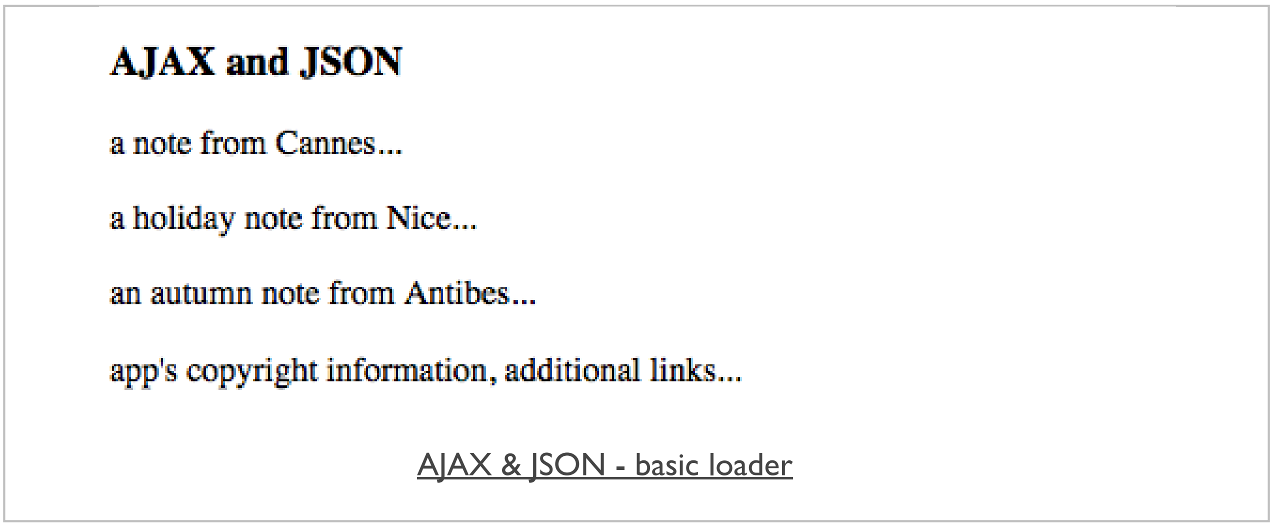
example - work with data

- returned data
  - *our response returns an object containing an array with notes*
- we could simply extract the required notes
  - *then append them to the DOM*

```
$deferredNotesRequest.done(function(response) {
    //get travelNotes
    var $travelNotes = response.travelNotes
    //process travelNotes array
    $travelNotes.forEach(function(item) {
      if (item !== null) {
        var note = item.note;
        //create each note's <p>
        var p = $("<p>");
        //add note text
        p.html(note);
        //append to DOM
        $(".note-output").append(p);
      }
    });
});
```

- DEMO - ajax & json basic loader

Image - HTML5, CSS, & JS - AJAX & JSON



AJAX and JSON

a note from Cannes...

a holiday note from Nice...

an autumn note from Antibes...

app's copyright information, additional links...

AJAX & JSON - basic loader

Ajax, JSON & jQuery - part 10

example - work with data

- we can use simple deferred requests with our local JSON data
- with staggered API calls to data, need to use slightly modified approach
    - *digging through data layer by layer*
    - *submitting a requestas one layer returns*
- we could now create a second deferred object
    - *use to track additional processing requests*
    - *stagger our requests to the API*
    - *ensuring we only request certain data as needed or available*
- also create multiple deferred objects to handle our requests and returned data
    - *allows us to respond accordingly within the application*

Ajax, JSON & jQuery - part 11

example - work with data

`resolve()`

- use this method with the deferred object to change its state, effectively to complete
- as we resolve a deferred object
    - *any **doneCallbacks** added with `then()` or `done()` methods will be called*
    - *these callbacks will then be executed in the order added to the object*
    - *arguments supplied to `resolve()` method will be passed to these callbacks*

`promise()`

- useful for limiting or restricting what can be done to the deferred object

```
function returnPromise() {
  return $.Deferred().promise();
}
```

- method returns an object with a similar interface to a standard `deferred` object
    - *only has methods to allow us to attach callbacks*
    - *does not have the methods required to resolve or reject deferred object*
- restricting the usage and manipulation of the deferred object
    - *eg: offer an API or other request the option to subscribe to the deferred object*
    - ***NB:** they won't be able to resolve or reject it as standard*

Ajax, JSON & jQuery - part 12

example - work with data

- still use the `done()` and `fail()` methods as normal
- use additional methods with these callbacks including the `then()` method
- use this method to return a new promise
    - *use to update the status and values of the deferred object*
    - *use this method to modify or update a deferred object as it is resolved, rejected, or still in use*
- can also combine promises with the `when()` method
    - *method allows us to accept many promises, then return a sort of master deferred*
- updated `deferred` object will now be resolved when all of the promises are resolved
    - *it will likewise be rejected if any of these promises fail*
- use standard `done()` method to work with results from all of the promises
    - *eg: could use this pattern to combine results from multiple JSON files*
    - *multiple layers within an API*
    - *staggered calls to paged results in a API...*

Ajax, JSON & jQuery - part 13

example - work with data

- now start to update our test AJAX and JSON application
  - *begin by simply abstracting our code a little*

```
function buildNote(data) {
  //create each note's <p>
  var p = $("<p>");
  //add note text
  p.html(data);
  //append to DOM
  $(".note-output").append(p);
}

//get the notes JSON
function getNotes() {
  //.get returns an object derived from a Deferred object - do not need explicit deferred
  var $deferredNotesRequest = $.getJSON (
  "docs/json/notes.json",
  {format: "json"}
  );
  return $deferredNotesRequest;
}
```

- DEMO - ajax & json abstract loader

Ajax, JSON & jQuery - part 14

example - work with data

- requesting our JSON file using `.getJSON()`
  - *we get a returned **promise** for the data*
- with a **promise** we can only use the following
  - *deferred object's method required to attach any additional handlers*
  - *or determine its state*
- our **promise** can work with
  - *`then`, `done`, `fail`, `always`...*
- our **promise** can't work with
  - *`resolve`, `reject`, `notify`...*

Ajax, JSON & jQuery - part 15

example - work with data

- one of the benefits of using **promises** is the ability to load one JSON file
  - *then wait for the results*
  - *then issue a follow-on request to another file*
  - ...
- a simple example of chained `then()` methods

```
getNotes().then(function(response1) {
  console.log("response1="+response1.travelNotes[2].note);
  $(".note-output").append(response1.travelNotes[2].note);
  return getPlaces();
}).then(function(response2) {
  console.log("response2="+response2.travelPlaces[2].place);
  $(".note-output").append(response2.travelPlaces[2].place);
});
```

- outputting a limited test result to the DOM and the console
- as we chain our `then()` methods
  - *pass returned results to next chained `then()` method...*
- DEMO - ajax & json deferred .then()

HTML5, CSS, & JS - example - part 1

add AJAX and JSON - load notes from json

- update our **travel notes** app to allow us to load some test persistent notes from a local JSON file
- initial JSON is as follows

```
{
  "travelNotes": [{
    "created": "2015-10-12T00:00:00Z",
    "note": "a note from Cannes..."
  }, {
    "created": "2015-10-13T00:00:00Z",
    "note": "a holiday note from Nice..."
  }, {
    "created": "2015-10-14T00:00:00Z",
    "note": "an autumn note from Antibes..."
  }]
}
```

HTML5, CSS, & JS - example - part 2

add AJAX and JSON - load notes from json

- add option to load notes from JSON as app initially loads
  - *use deferred promise pattern*
  - *checks source JSON as it loads via the promise*
  - *then outputs the end result*
- start with the following update

```
//get the notes JSON
function getNotes() {
  //.get returns an object derived from a Deferred object - do not need explicit deferred
  var $deferredNotesRequest = $.getJSON (
    "docs/json/notes.json",
    {format: "json"}
  );
  return $deferredNotesRequest;
}
```

HTML5, CSS, & JS - example - part 3

add AJAX and JSON - load notes from json

- help us better manage logic of our notes from app's loading and execution
  - *create two separate JS files*
- our updated structure might be as follows

```
...
|- assets
   |- scripts
      |- travel.js
      |- notes.js
...
```

- we can extend this further, as needed by app features and data

HTML5, CSS, & JS - example - part 4

add AJAX and JSON - load notes from json

- add our `.when()` function to the app's loader
  - *`.when()` function accepts a deferred object*
  - *in our case a limited promise*
- then allows us to chain additional deferred functions
  - *including required `.done()` function*
- for returned data, use standard response object to get `travelNotes`
  - *then iterate over the array for each property*
  - *for each iteration, we can simply call our `createNote` function*
  - *builds and renders required notes to the app's DOM*

```
//use deferred object from getJson
$.when(getNotes()).done(function(response) {
  //get travelNotes object
  var $travelNotes = response.travelNotes
  //process travelNotes array
  $travelNotes.forEach(function(item) {
    //check each property
    if (item !== null) {
      //get note
      var note = item.note;
      //create each note for rendering
      createNote(note);
    }
  });//end foreach
});
```

HTML5, CSS, & JS - example - part 5

add AJAX and JSON - load notes from json

- simple problem - existing `createNote()` function does not accept a parameter
- need to update the logic of that function to accept and handle a parameter
- also requires a quick update to any functions and calls to the `createNote()`
    - *event handlers for creating a new note using the `add` button and keypress within the input field*

```
//manage input field and new note output
function createNote(data) {
  ...
  //conditional check for data
  if (data !== "") {
  //set content for note
  $note.html(data);
  ...
  }
}
```

HTML5, CSS, & JS - example - part 6

add AJAX and JSON - load notes from json

- update our event handlers for the note input button and input field keypress as follows,

```
//handle user event for `add` button click
$(".note-input button").on("click", function(e) {
  var $note_data = getNoteInput();
  //call note builder function
  createNote($note_data);
});
```

```
//handle user event for keyboard press
$(".note-input input").on("keypress", function(e) {
  //check code for keyboard press
  if (e.keyCode === 13) {
    var $note_data = getNoteInput();
    //call note builder function
    createNote($note_data);
  }
});
```

- our notes now load by default as the app starts
- note input button and keypress work as expected
- DEMO - travel notes & JSON

Working with APIs - part 1

remote api options - Flickr

- **Travel Notes** app loads data from a local JSON file
- add option to load different types of data using remote APIs
  - *Flickr API for images, tags...*
- basics and principles are similar to the patterns we've already seen and tested
- test a sample JSON return from the Flickr API
- JSON return - useful properties for app
  - *title*
  - *link*
  - *media (direct url for image - where available)*
  - *description*
  - *...*
- public feed for searching public photos, videos, groups, recent activity...
- Flickr API Public Feed - Cannes and France

Working with APIs - part 2

working with Flickr API

- query Flickr's public feed for photos
  - *we can use our now familiar pattern for requesting JSON*

```
//get the Flickr public feed JSON for images
function getImages() {
  //.get returns an object derived from a Deferred object - do not need explicit deferred
  var $deferredNotesRequest = $.getJSON (
  "http://api.flickr.com/services/feeds/photos_public.gne?jsoncallback=?",
    { tags: "cannes,france,boules",
      tagmode: "all",
      format: "json"
    });
  return $deferredNotesRequest;
}
```

- need to make a few specific modifications to the request
  - *JSONP to avoid browser security restrictions*

Working with APIs - part 3

working with Flickr API

- Flickr's public feed includes options
    - *eg: a specific user ID for photos, various tags, how tags are interpreted by the search...*
- use our `.when()` function to load and render some test images from Flickr

```
$.when(getImages()).done(function(response) {
  console.log("done..."+response);
  //use jQuery's generic iterative function for the response...
  $.each( response.items, function( i, item ) {
    buildImage(item.media.m);
    //limit test images to 8
    if ( i === 7 ) {
      return false;
    }
  });
});
```

- DEMO - AJAX and JSON - Flickr api

HTML5, CSS, & JS - example - part 7

working with Flickr API - update travel notes

- add option to Travel Notes app to allow a user to view images from Flickr
- need to update app's HTML, CSS, and JS
- modify how our notes, and associated options, are rendered to our users
- add a search option for photos on Flickr
- render our images to match the notes
- app's structure still reflects three primary content categories
  - *`header`, `main`, and `footer` with slight modifications to the `main` category*
- `main` content category updated to create two distinct rows for initial content
  - *contain defined semantic containers*
- row containing `.note-input` and Flickr search option `.contextual-choice`
  - *then split this row into two columns of 6*

HTML5, CSS, & JS - example - part 8

working with Flickr API - update travel notes HTML

- updated HTML for `.note-input` and Flickr search `.contextual-choice`

```
<div class="row">
  <!-- note input -->
  <section class="note-input col-6">
    <h5>add note</h5>
    <input><button>add</button>
  </section>
  <!-- contextual choice -->
  <section class="contextual-choice col-6">
    <h5>search flickr</h5>
    <input><button>search</button>
  </section>
</div>
```

HTML5, CSS, & JS - example - part 9

working with Flickr API - update travel notes HTML

- update the HTML for rendering the images
    - *add alongside our notes*
- create another row for these containers
    - *add two section containers for `.note-output` and `.contextual-output`*
- make `.note-output` slightly larger to show primary app focus

```html
<div class="row">
  <!-- note output -->
  <section class="note-output col-7 flex-container">
  </section>
  <!-- contextual output -->
  <section class="contextual-output col-5 flex-container">
  </section>
</div>
```

HTML5, CSS, & JS - example - part 10

working with Flickr API - update travel notes JS

- add further functionality to **Travel Notes** app
- split our JS logic into three files to help with oranisation
  - *a main loader file, `travel.js`,*
  - *and a file each for notes and contextual options
- updated app structure for JS

```
...

|- assets
   |- scripts
      |- contextual.js
      |- notes.js
      |- travel.js
...
```

- underlying logic for the notes will remain the same
  - *move loading of default notes to the `travel.js` main loader file*
- updates for searching, returning, and rendering images from Flickr
  - *added to the `contextual.js` file*

HTML5, CSS, & JS - example - part 11

working with Flickr API - update travel notes JS

- test Flickr API in our app using some set data for image tags
  - *respond to the user clicking on the search button*
  - *submit our query to Flickr*
  - *process the returned JSON for the images*
  - *render them for viewing*
- request and process our images using the familiar pattern

```
//get the Flickr public feed JSON for images
function getImages(data) {
  var img_tags = data;
  //.get returns an object derived from a Deferred object - do not need explicit deferred
  var $deferredNotesRequest = $.getJSON (
    "http://api.flickr.com/services/feeds/photos_public.gne?jsoncallback=?",
    { tags: img_tags,
      tagmode: "all",
      format: "json"
    });
  return $deferredNotesRequest;
}
```

HTML5, CSS, & JS - example - part 12

working with Flickr API - update travel notes JS

- returned data using standard deferred promise object
  - *add a new function to handle the processing of the images*

```
function processImages(data) {
  $.when(getImages($img_data)).done(function(response) {
    //use jQuery's generic iterative function for the response...
    $.each( response.items, function( i, item ) {
      createImage(item.media.m);
      //limit test images to 4
      if ( i === 3 ) {
        return false;
      }
    });
  });
}
```

- using deferred promise object with `.when()` function chained to `.done()` function
- add jQuery's generic iterative function to help us process the response
  - *instead of standard JavaScript `.forEach()` option*
- loop through each value, and pass the image to our new function, `createImage()`
  - *ready for rendering to our app's DOM*
  - *limit number of images for testing*

HTML5, CSS, & JS - example - part 13

working with Flickr API - update travel notes JS

```
//manage new image output
function createImage(data) {
  //create each image element
  var img = $('<img class="flex-img">');
  //add image
  img.attr("src", data);
  //append to DOM
  $(".contextual-output").append(img);
}
```

- `.createImage()` function accepts a parameter for image data
- then process ready for rendering to the app's DOM
- image is added to a new `img` element with a new class of `.flex-img`
  - *creates a flex item for rendering*
- added to the new `.contextual-output` section
- rendered images displayed as thumbnails for the user
  - *complementary to the existing notes*

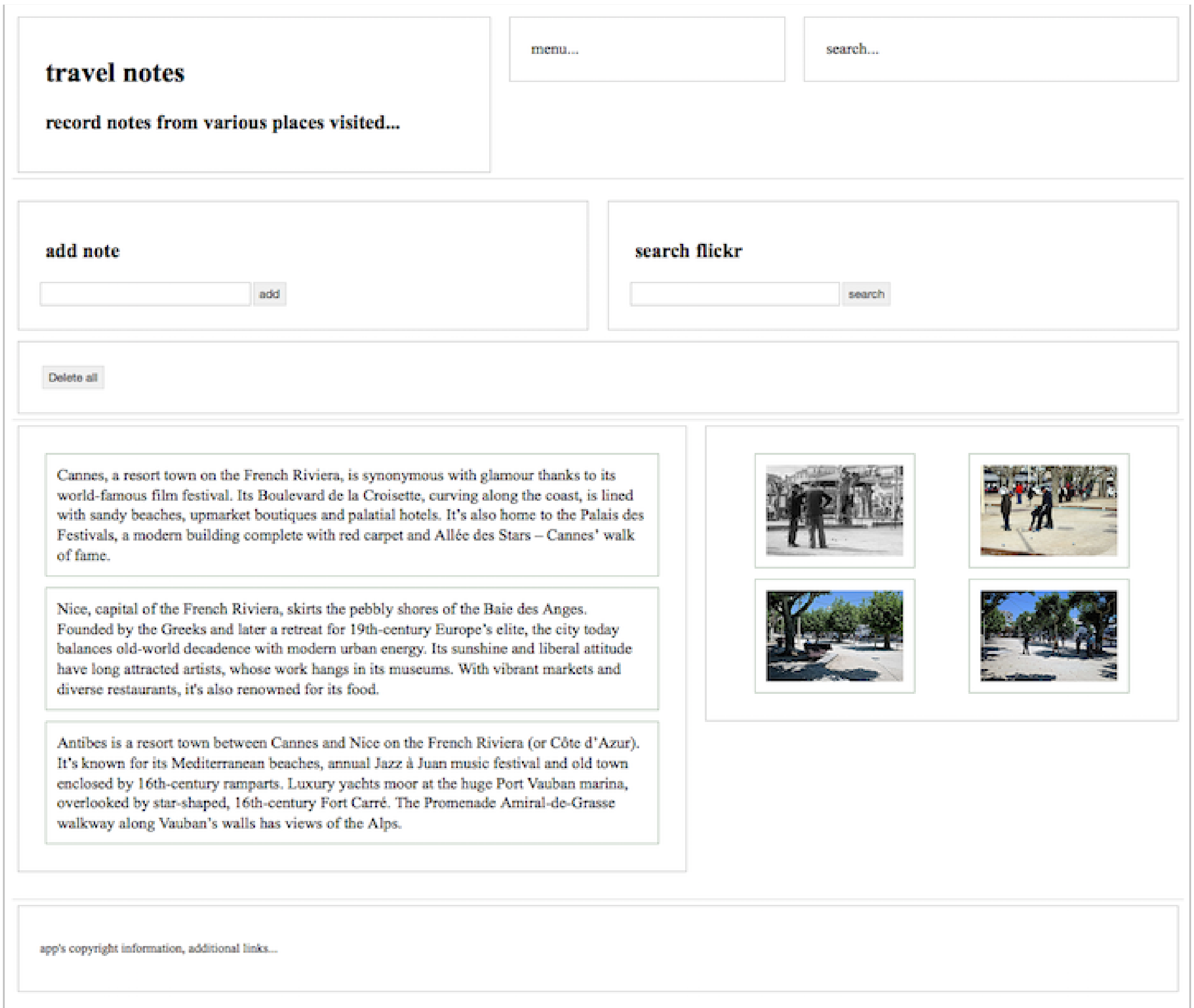
HTML5, CSS, & JS - example - part 14

working with Flickr API - update travel notes JS

- to add images to the app
  - *a user can enter their requested tags in the search field*
  - *then click on the `search` button to return any available images*
- event handler for this `search` button click uses the requested tags
  - *passes them as a parameter to the `processImages()` function*

```
//handle user event for image `search` button click
$(".contextual-choice button").on("click", function(e) {
  //test tags for testing image search
  $img_data = "cannes,france,boules"
  //process images
  processImages($img_data);
});
```

Image - HTML5, CSS, & JS - Travel Notes & Flickr



travel notes

record notes from various places visited...

Cannes, a resort town on the French Riviera, is synonymous with glamour thanks to its world-famous film festival. Its Boulevard de la Croisette, curving along the coast, is lined with sandy beaches, upmarket boutiques and palatial hotels. It's also home to the Palais des Festivals, a modern building complete with red carpet and Allée des Stars – Cannes' walk of fame.

Nice, capital of the French Riviera, skirts the pebbly shores of the Baie des Anges. Founded by the Greeks and later a retreat for 19th-century Europe's elite, the city today balances old-world decadence with modern urban energy. Its sunshine and liberal attitude have long attracted artists, whose work hangs in its museums. With vibrant markets and diverse restaurants, it's also renowned for its food.

Antibes is a resort town between Cannes and Nice on the French Riviera (or Côte d'Azur). It's known for its Mediterranean beaches, annual Jazz à Juan music festival and old town enclosed by 16th-century ramparts. Luxury yachts moor at the huge Port Vauban marina, overlooked by star-shaped, 16th-century Fort Carré. The Promenade Amiral-de-Grasse walkway along Vauban's walls has views of the Alps.

app's copyright information, additional links...

Travel Notes & Flickr - test loading images

HTML5, CSS, & JS - example - part 15

working with Flickr API - update travel notes CSS

- need to update and modify existing CSS
  - *helps with correct rendering of the thumbnail images*
- CSS additions are initially modest
  - *reflects integration with existing app, grid, and flex layouts*
- add new ruleset for image rendering in the `.contextual-output` section

```css
/* contextual output images */
.contextual-output img {
  margin: 5px;
  padding: 5px;
  border: 1px solid #b1c4b1;
}
```

- update `.flex-container` class to change `justify-content` property to value of `space-around`
- add new ruleset for a `.flex-img` class.

```css
/* flex image */
.flex-img {
  flex-basis: 150px;
  flex-grow:0;
}
```

- specify size of a thumbnail image
  - *initially restrict their ability to grow relative to flex*

HTML5, CSS, & JS - example - part 16

working with Flickr API - update travel notes JS

- we can now request, process, and render images from Flickr to Travel Notes app
  - *still need to accept and process search queries from search input field.*
- add option to check search input field
  - *then submit query to Flickr for images*

```
//get input value for image search
function getImageInput() {
  //define img value
  var img_val = "";
  //define input field
  var $img_tags = $(".contextual-choice input");
  if ($img_tags.val() !== "") {
    img_val = $img_tags.val();
    return img_val;
  } else {
    return img_val;
  }
}
```

HTML5, CSS, & JS - example - part 17

working with Flickr API - update travel notes JS

- use `getImageInput()` function with a modified `processImages()` function

```
//process image production, loading, and pass to rendering
function processImages() {
  //check img visibility for contextual-output - clear existing images
  if (checkVisible($(".contextual-output img")) === false) {
    //empty existing images
    $(".contextual-output").empty();
  }
  //get data from image search input field
  var $img_data = getImageInput();
  //use image data to get images, and pass for rendering
  $.when(getImages($img_data)).done(function(response) {
    console.log("done..."+response);
    //use jQuery's generic iterative function for the response...
    $.each( response.items, function( i, item ) {
      createImage(item.media.m);
      //limit test images to 4
      if ( i === 3 ) {
        return false;
      }
    });
  });
}
```

HTML5, CSS, & JS - example - part 18

working with Flickr API - update travel notes JS

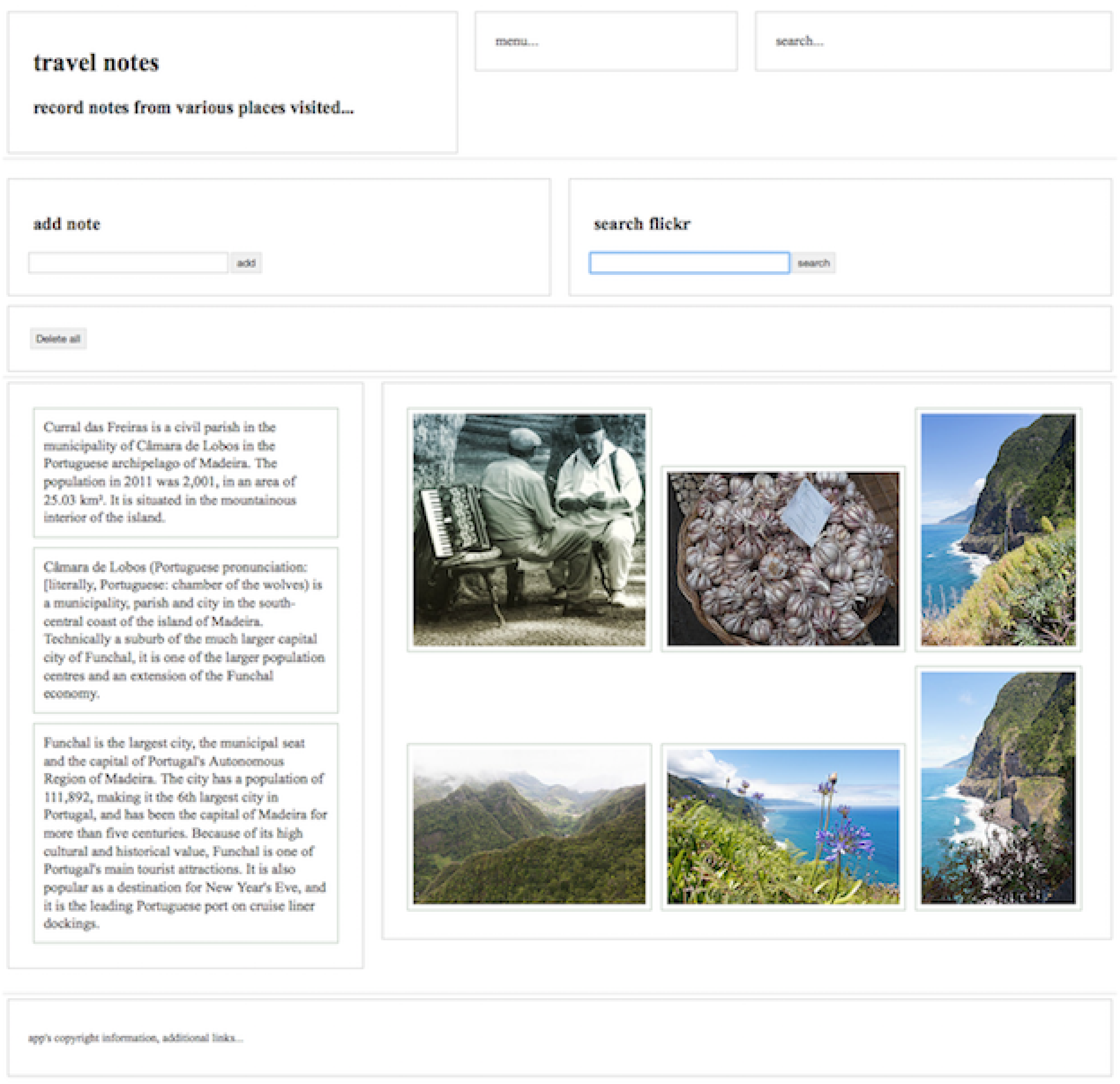
- updated `processImages()` function then called within event handlers
  - *for the search button and a keypress in the search input field*

```
//handle user event for image search button click
$(".contextual-choice button").on("click", function(e) {
  //process images
  processImages();
});

//handle user event for keyboard press
$(".contextual-choice input").on("keypress", function(e) {
  //check code for keyboard press
  if (e.keyCode === 13) {
    //process images
    processImages();
  }
});
```

- DEMO - travel notes & Flickr

Image - HTML5, CSS, & JS - Travel Notes & Flickr



Curral das Freiras is a civil parish in the municipality of Câmara de Lobos in the Portuguese archipelago of Madeira. The population in 2011 was 2,001, in an area of 25.03 km². It is situated in the mountainous interior of the island.

Câmara de Lobos (Portuguese pronunciation: [literally, Portuguese: chamber of the wolves) is a municipality, parish and city in the south-central coast of the island of Madeira. Technically a suburb of the much larger capital city of Funchal, it is one of the larger population centres and an extension of the Funchal economy.

Funchal is the largest city, the municipal seat and the capital of Portugal's Autonomous Region of Madeira. The city has a population of 111,892, making it the 6th largest city in Portugal, and has been the capital of Madeira for more than five centuries. Because of its high cultural and historical value, Funchal is one of Portugal's main tourist attractions. It is also popular as a destination for New Year's Eve, and it is the leading Portuguese port on cruise liner dockings.

Travel Notes & Flickr - different layout

HTML5, CSS, & JS - example - part 19

working with Flickr API - update travel notes JS

- room for improvement, updates, abstraction, and general refactoring of the existing code
- return to this issue when we consider refactoring the code in general
  - *there are still a few simple features we need to add*
- for example,
  - *add images to the `.contextual-output` section, resize `.note-output` section*
  - *moves focus to the current images*
  - *check loading progress of the notes and images*
  - *show feedback to the user*
  - *need to output a title for the images*
  - *set using the search query*

HTML5, CSS, & JS - example - part 20

working with Flickr API - modify travel notes JS

- first modification is to resize the `.notes-output`
  - *create more space for the images*
  - *gently shift focus to the new images*
- update existing `.createImage()` function in the `contextual.js` file

```
//manage new image output
function createImage(data) {
...
    if (checkVisible($(".contextual-output img")) === true) {
      $(".note-output").removeClass("col-12");
      $(".note-output").addClass("col-4");
      $(".contextual-output").fadeIn("slow");
    }
...
}
```

- add check to ensure images are not visible in the DOM
- remove current class from `.note-output` section
  - *12 column class for the grid*
- add new grid class to resize `.note-output` to 4 columns
  - *then fade in the `.contextual-output` class*
  - *set in the app's HTML to a class of `.col-8`*

HTML5, CSS, & JS - example - part 21

working with Flickr API - modify travel notes JS

- next modification is some initial error handling
  - *checking for an empty array of images from the returned Flickr JSON*
- check `processImages()` function for an empty array of image items

```
...

if (response.items.length === 0) {
  var img = "";
  createImage(img);
} else {
  //return images from items array...
}
...
```

- checks images in the items array for the promise object
- if not, send an empty variable as a parameter to our `createImage()` function

HTML5, CSS, & JS - example - part 22

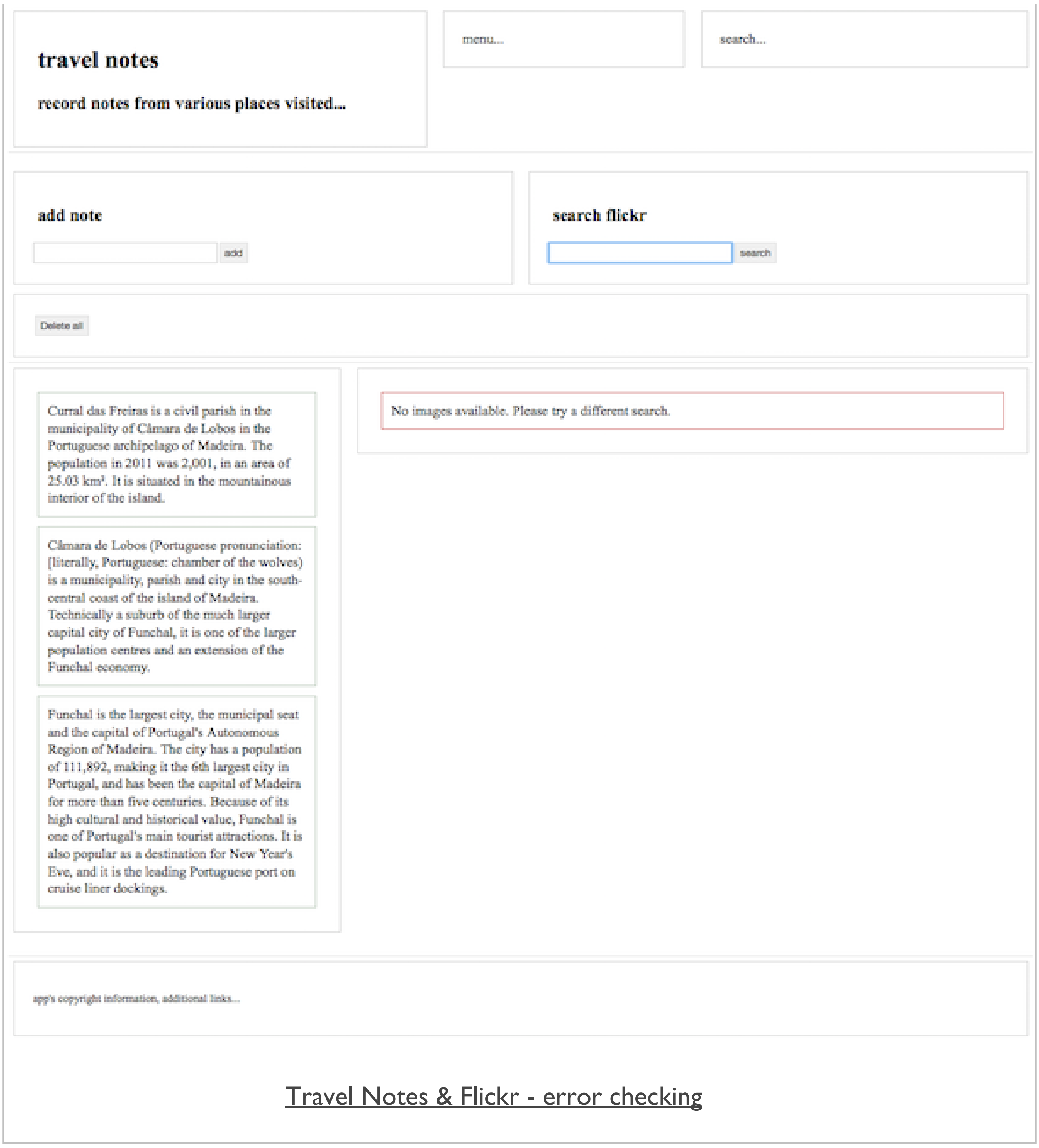
working with Flickr API - modify travel notes JS

- check for empty value in `createImage()` function
  - *handle the simple errors as follows*

```
if (data !== "") {
  //create each image element
  var $img = $('<img class="flex-img">').attr("src", data);
  //add image
  img_output = $img;
} else {
  var $img_error = $('<p class="flex-item error">').html("No images available...");
  //add error
  img_output = $img_error;
}
```

- we've abstracted the return variable for the image output
  - *can hold either the image or the error output...*
- add a check to see whether the `.contextual-output` section is visible or not
- modify the column class for the `.note-output` section
- then append our image output
- then show the `.contextual-output` section within the app
- DEMO - travel notes & Flickr

Image - HTML5, CSS, & JS - Travel Notes & Flickr



travel notes

record notes from various places visited...

add note

search flickr

Curral das Freiras is a civil parish in the municipality of Câmara de Lobos in the Portuguese archipelago of Madeira. The population in 2011 was 2,001, in an area of 25.03 km². It is situated in the mountainous interior of the island.

Câmara de Lobos (Portuguese pronunciation: [literally, Portuguese: chamber of the wolves) is a municipality, parish and city in the south-central coast of the island of Madeira. Technically a suburb of the much larger capital city of Funchal, it is one of the larger population centres and an extension of the Funchal economy.

Funchal is the largest city, the municipal seat and the capital of Portugal's Autonomous Region of Madeira. The city has a population of 111,892, making it the 6th largest city in Portugal, and has been the capital of Madeira for more than five centuries. Because of its high cultural and historical value, Funchal is one of Portugal's main tourist attractions. It is also popular as a destination for New Year's Eve, and it is the leading Portuguese port on cruise liner dockings.

No images available. Please try a different search.

app's copyright information, additional links...

Travel Notes & Flickr - error checking

HTML5, CSS, & JS - example - part 23

working with Flickr API - modify travel notes JS

- continue to modify and build our Travel Notes app
- add some metadata for the returned images
  - *using the title and link from the search query response*
- add initial metadata output in the `contextual.js` file
  - *modify the `processImages()` function*
  - *metadata from Flickr JSON response in the deferred promise object*

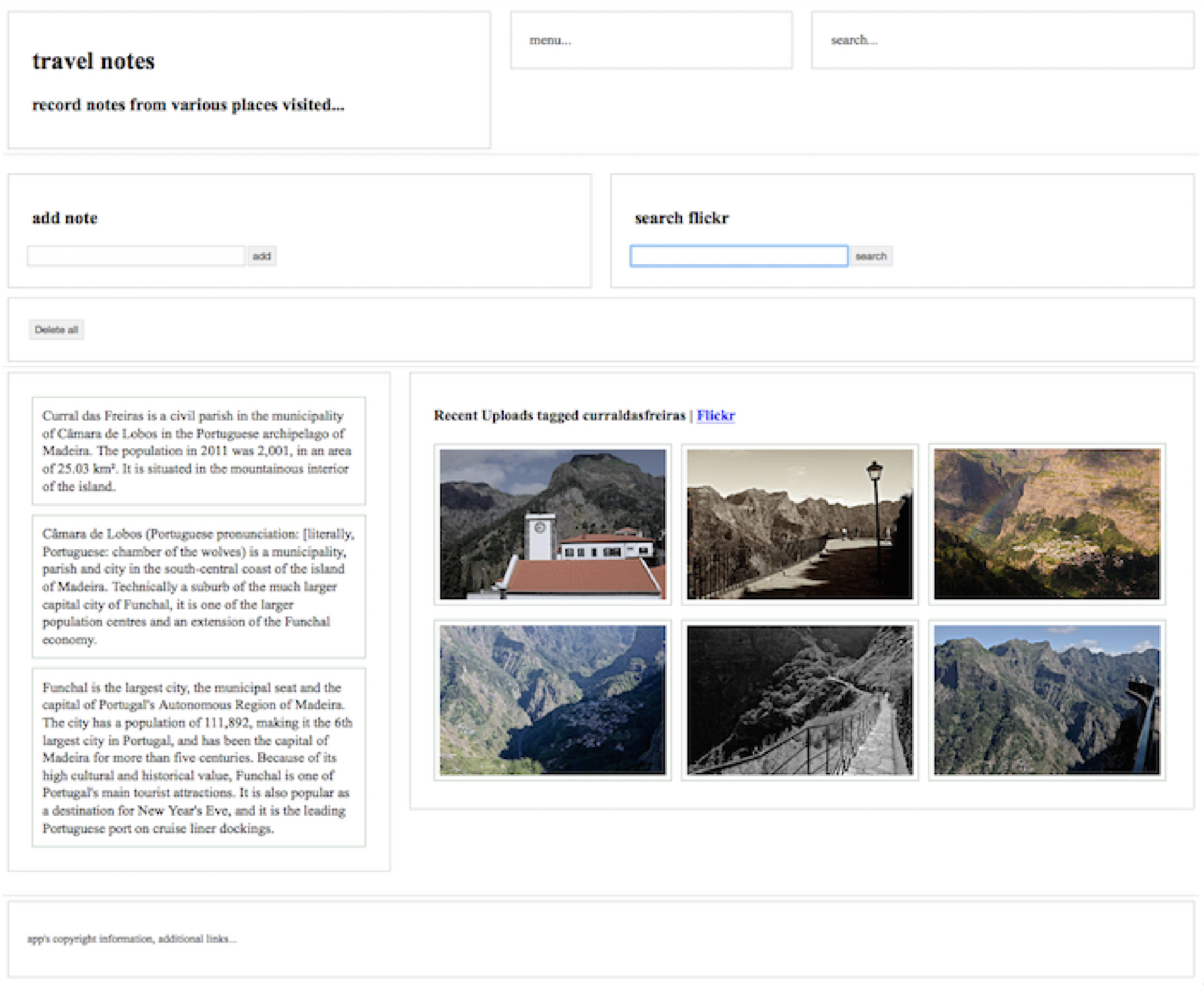
```
...
//create object for search metadata
var search_meta = {title:response.title, link:response.link};
...
```

- then pass this to a new function, called `metaOutput()`

```
//prepare and render metadata for returned search...
function metaOutput(data) {
  if (data !== "") {
  //search metadata from response
  var search_title = data.title;
  var search_link = data.link;
  //build heading output for metadata heading
  var metaHeading = '<h6>'+search_title+' | <a href="'+search_link+'">Flickr</a></h6>';
  //render metadata to contextual-output
  $(".contextual-output").prepend(metaHeading);
  }
}
```

- DEMO - travel notes & Flickr - initial metadata

Image - HTML5, CSS, & JS - Travel Notes & Flickr



travel notes

record notes from various places visited...

menu...

search...

add note

search flickr

Curral das Freiras is a civil parish in the municipality of Câmara de Lobos in the Portuguese archipelago of Madeira. The population in 2011 was 2,001, in an area of 25.03 km². It is situated in the mountainous interior of the island.

Câmara de Lobos (Portuguese pronunciation: [literally, Portuguese: chamber of the wolves) is a municipality, parish and city in the south-central coast of the island of Madeira. Technically a suburb of the much larger capital city of Funchal, it is one of the larger population centres and an extension of the Funchal economy.

Funchal is the largest city, the municipal seat and the capital of Portugal's Autonomous Region of Madeira. The city has a population of 111,892, making it the 6th largest city in Portugal, and has been the capital of Madeira for more than five centuries. Because of its high cultural and historical value, Funchal is one of Portugal's main tourist attractions. It is also popular as a destination for New Year's Eve, and it is the leading Portuguese port on cruise liner dockings.

Recent Uploads tagged curraldasfreiras | Flickr

app's copyright information, additional links...

Travel Notes & Flickr - initial metadata

HTML5, CSS, & JS - example - part 24

travel notes - basic refactoring of JS

- as we continue to add features and modify existing code
  - *may start to see unnecessary repetition and function calls in the code*
- eg: initial error handling for our contextual images
  - *`createImage()` function is being called in the `processImages()` function*
  - *called regardless of returned image data*
- `createImage()` is being used unnecessarily to manage the error handling
- move check to `processImages()` function
  - *then call function to render necessary error message*

```
function outputError(message) {
  var $img_error = $('<p class="flex-item error">').html(message);
  //check for visible contextual-output - if not visible
  if (checkVisible($(".contextual-output")) === true) {
    $(".note-output").removeClass("col-12");
    $(".note-output").addClass("col-4");
  }
  //append output to DOM
  $(".contextual-output").append($img_error);
  //fade in contextual-output with appended results
  $(".contextual-output").fadeIn("slow");
}
```

HTML5, CSS, & JS - example - part 25

travel notes - basic refactoring of JS

- updated `processImages()` function can call `.outputError()` function as needed

```
...
if (response.items.length !== 0) {
//logic to add metadata and each image...
}
else {
  var img_error = "No images available - please try a different search.";
  outputError(img_error);
}
...
```

- use this function to output error messages for any type of contextual data
- also remove some unnecessary replication of code
  - *by adding a simple function to change an element's class*

```
//modify element class - from, to
function changeClass(element, size1, size2) {
    $(element).removeClass(size1);
    $(element).addClass(size2);
}
```

- resize a class, for example to modify our grid output
  - *call this function - pass the selector to update, original class to remove, and new class to add*

HTML5, CSS, & JS - example - part 26

working with Flickr API - modify travel notes JS

- add a modification to check for the image loading and the notes
  - *offer status feedback to the user*

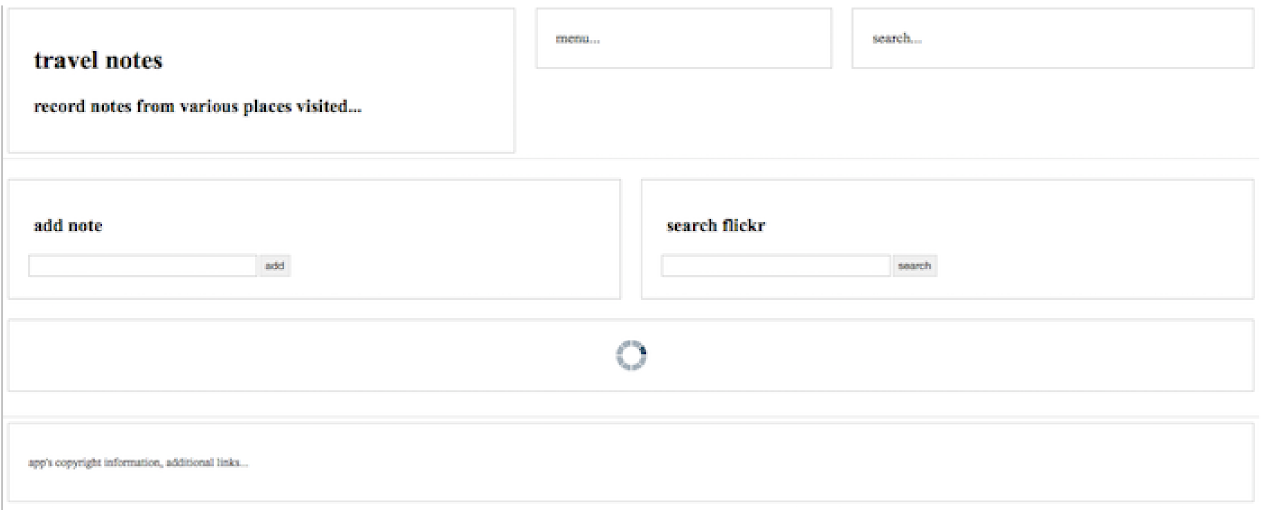
```
//add initial loader spinner for ajax...
$(".contextual-output").html('<img class="spinner" src="assets/images/ajax-loader.gif">');
```

- remove it when the deferred promise object has returned

```
//remove ajax spinner
$(".spinner").remove();
```

- DEMO - travel notes & Flickr - spinner

Image - HTML5, CSS, & JS - Travel Notes & Flickr


travel notes
record notes from various places visited...
menu...
search...
add note
add
search flickr
search
app's copyright information, additional links...

Travel Notes & Flickr - spinner

Demos

- AJAX and JSON
  - *AJAX-JSON 1 - load a JSON file*
  - *AJAX-JSON 2 - abstract code for load a JSON file*
  - *AJAX-JSON 3 - test deferred .then()*
  - *AJAX-JSON 4 - Flickr API*
- Travel Notes app
  - *DEMO 1 - Travel Notes & JSON*
  - *DEMO 2 - Travel Notes & Flickr*
  - *DEMO 3 - Travel Notes & Flickr - error checking*
  - *DEMO 4 - Travel Notes & Flickr - initial metadata*
  - *DEMO 5 - Travel Notes & Flickr - spinner*

References

- jQuery
  - *jQuery - deferred*
  - *jQuery - promise*
- Flickr API
  - *Public feeds*
  - *Public feed - public photos & video*
- Various
  - *Create your own AJAX loader*