Comp 424 - Client-side Web Design

Fall Semester 2016 - Week 11

Dr Nick Hayward

Contents

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

```
<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 object
  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*

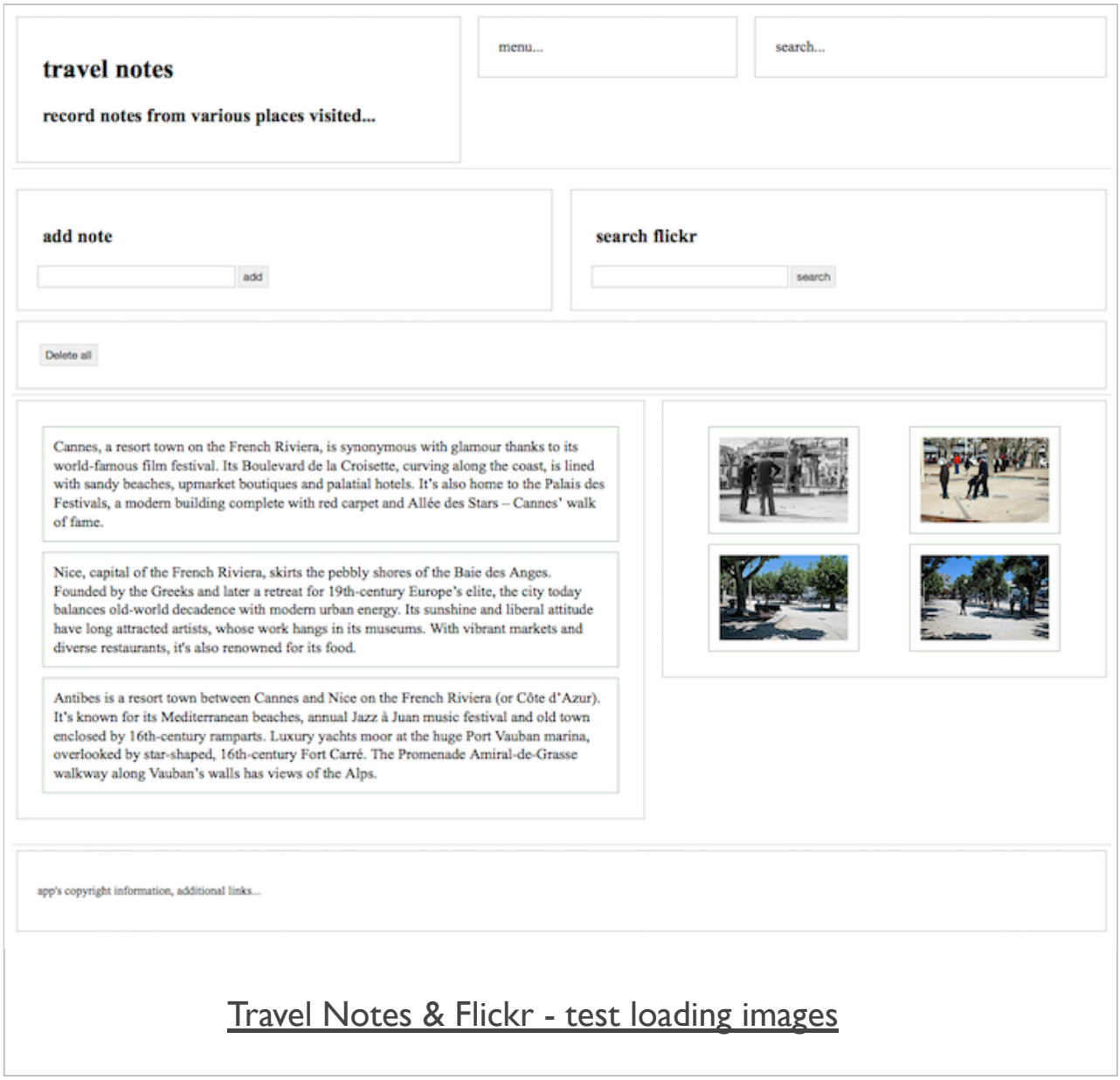
HTML5, CSS, & JS - example - part 14

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 & 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

```
/* 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.

```
/* 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

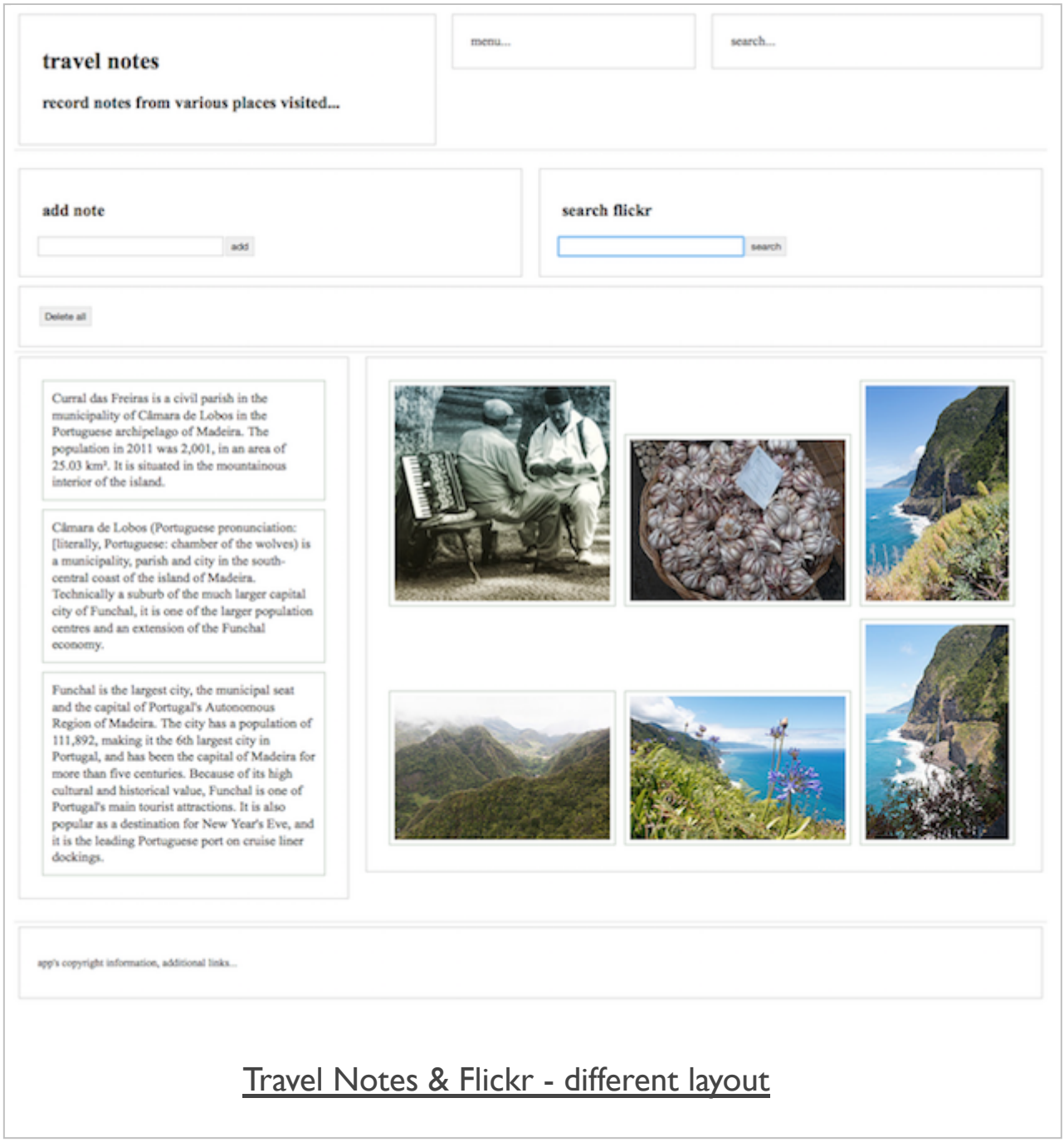
- updated `processImages()` function then called within event handlers
  - *for the search button and a keypress in the search input field*

```
//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



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

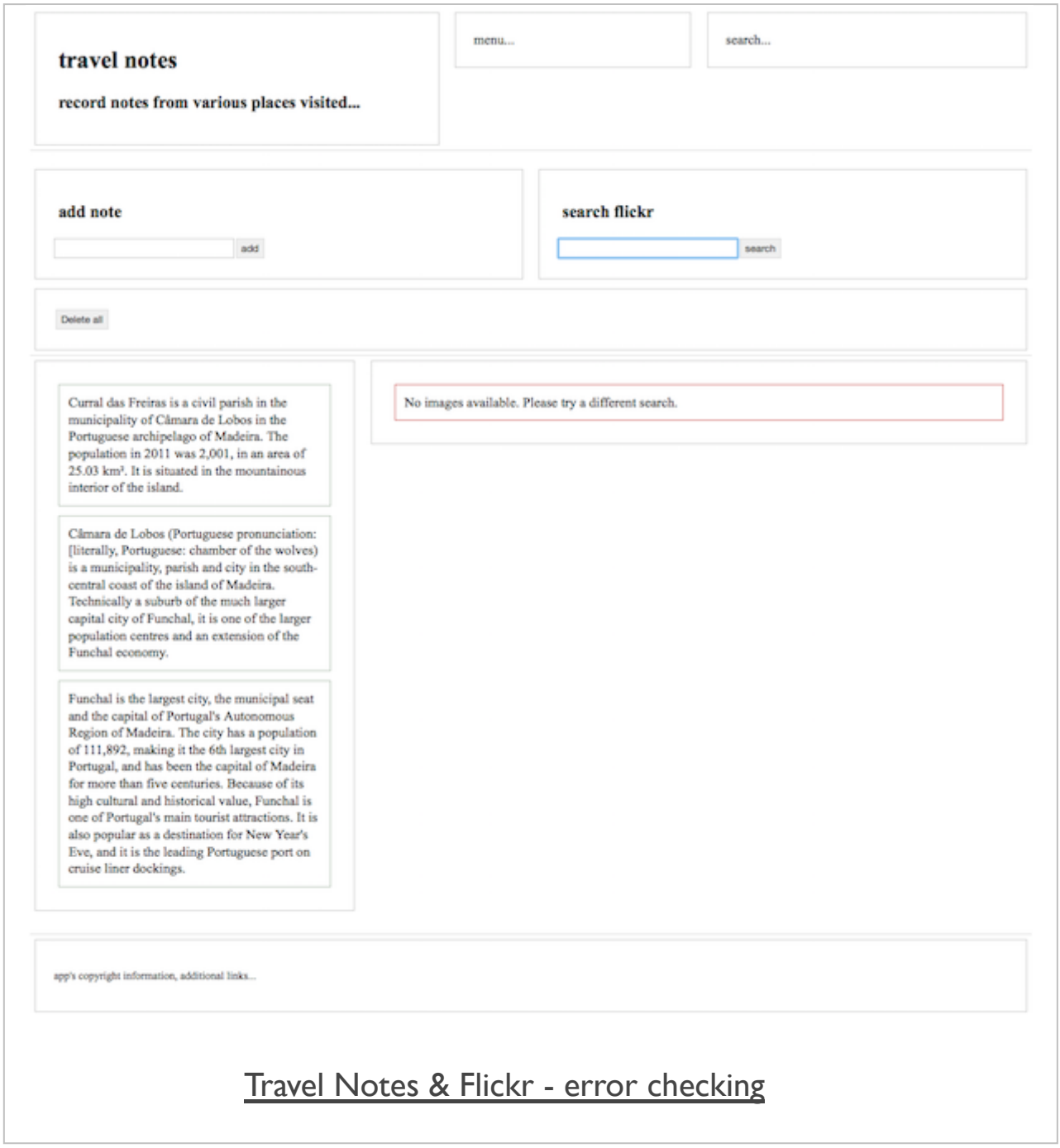
working with Flickr API - modify travel notes JS

- 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 & 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*

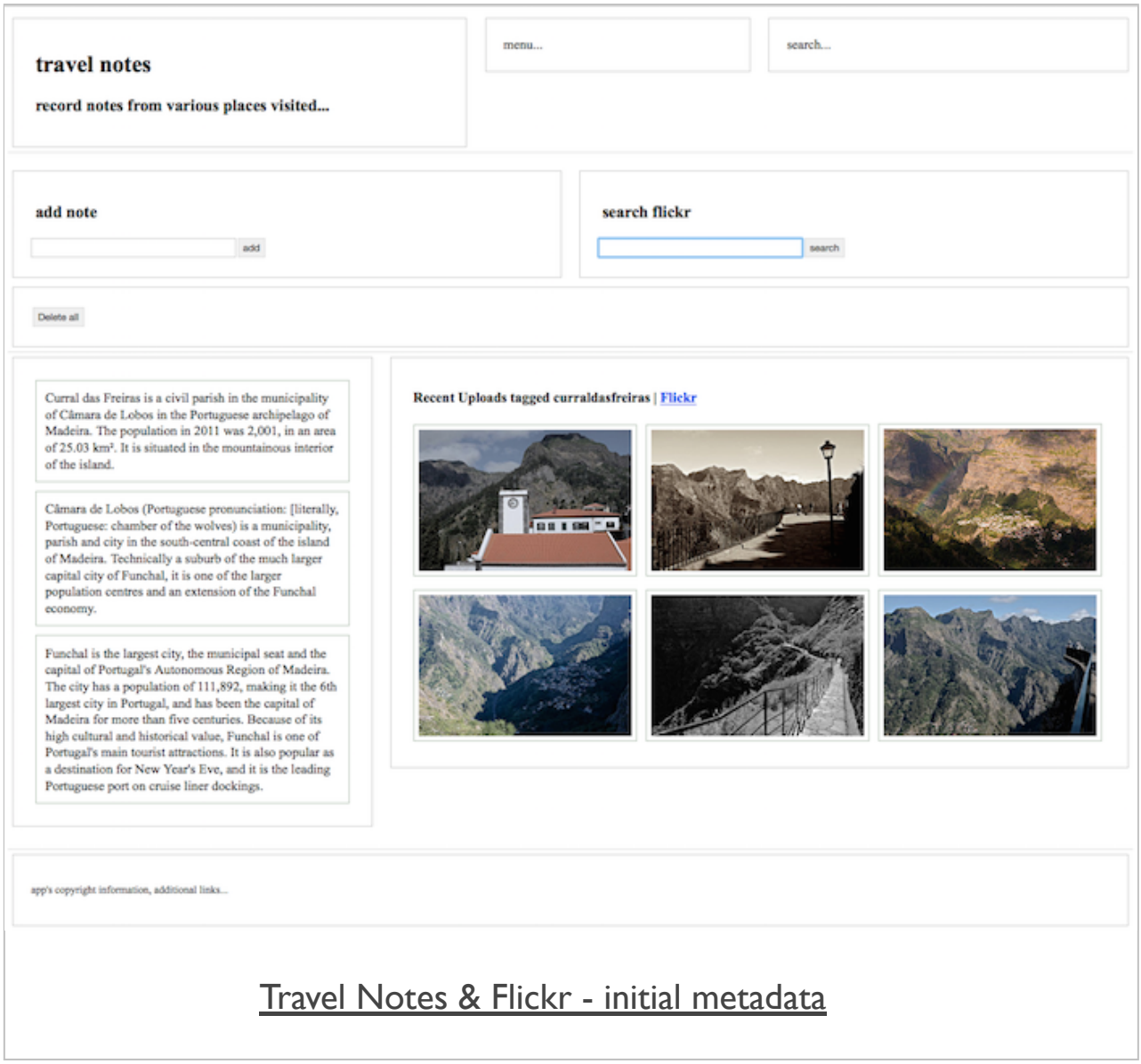
```
...
//create object for search metadata
var search_meta = {title:response.title, link:response.link};
...
```

- 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 & 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*

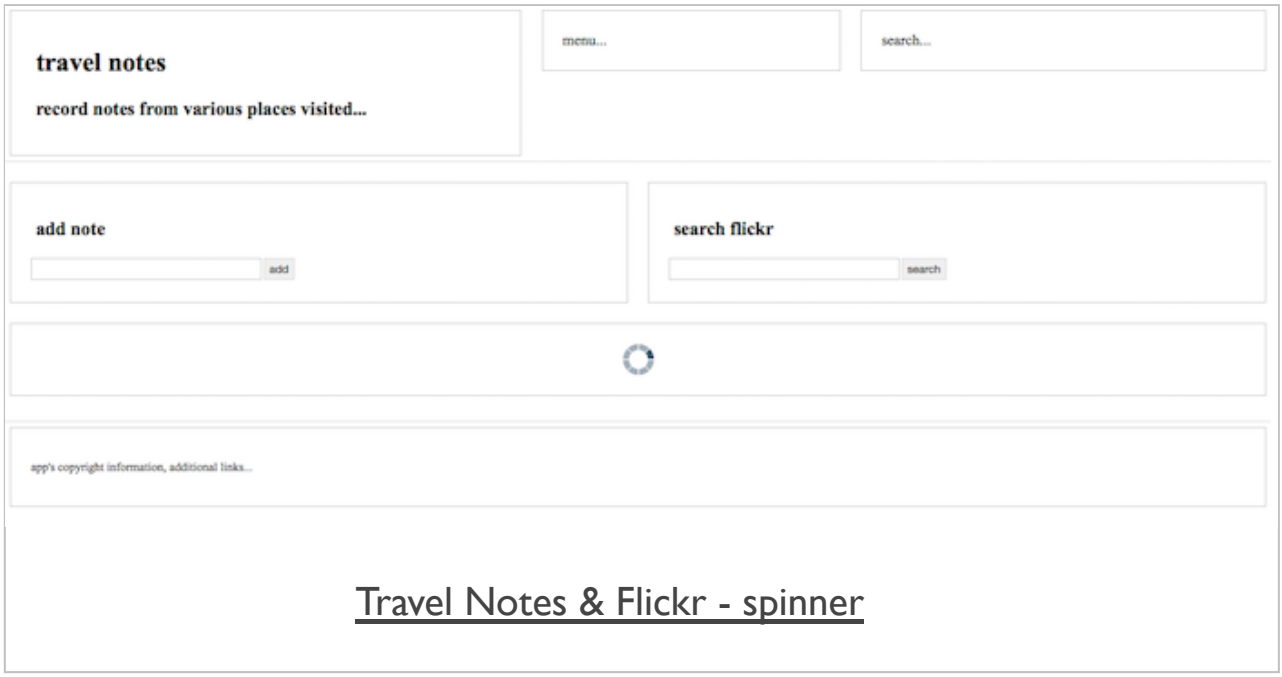
```
//add initial loader spinner for ajax...
$(".contextual-output").html('<img class="spinner" src="assets/images/ajax-loader.gif">');
```

- 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 & Flickr - spinner

JS Server-side considerations - save data

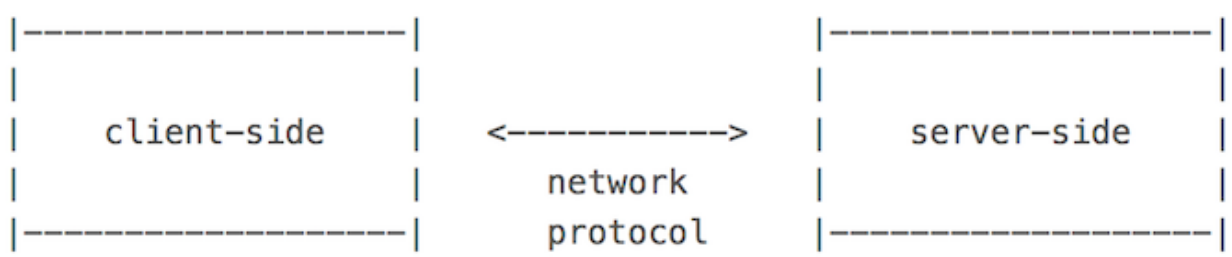
save JSON in travel notes app

- need to be able to save our simple notes
- now load from a JSON file as the app starts
  - *also we can add new notes, delete existing notes...*
- not as simple as writing to our existing JSON file direct from JS
  - *security implications if that was permitted directly from the browser*
- need to consider a few server-side options
- could use a combination of PHP on the server-side
  - *with AJAX jQuery on the client-side*
  - *traditional option with a simple ajax post to a PHP file on the server-side*
- consider JavaScript options on the client and server-side
- brief overview of working with **Node.js**

Server-side considerations - intro

- normally define computer programs as either client-side or server-side programs
- server-side programs normally abstract a resource over a network
  - *enabling many client-side programs to access at the same time*
  - *a common example is file requests and transfers*
- we can think of the client as the web browser
- a web server as the remote machine abstracting resources
- abstracts them via **hypertext transfer protocol**
  - *HTTP for short*
- designed to help with the transfer of HTML documents
  - *HTTP now used as an abstracted wrapper for many different types of resources*
  - *may include documents, media, databases...*

Image - Client-side and server-side computing



client-side & server-side

Server-side considerations - Node.js

intro - what is Node.js?

- Node.js is, in essence, a JavaScript runtime environment
  - *designed to be run outside of the browser*
- designed as a general purpose utility
- can be used for many different tasks including
  - *asset compilation*
  - *monitoring*
  - *scripting*
  - *web servers*
- with Node.js, role of JS is changing
  - *moving from client-side to a support role in back-end development*

Server-side considerations - Node.js

intro - speed of Node.js

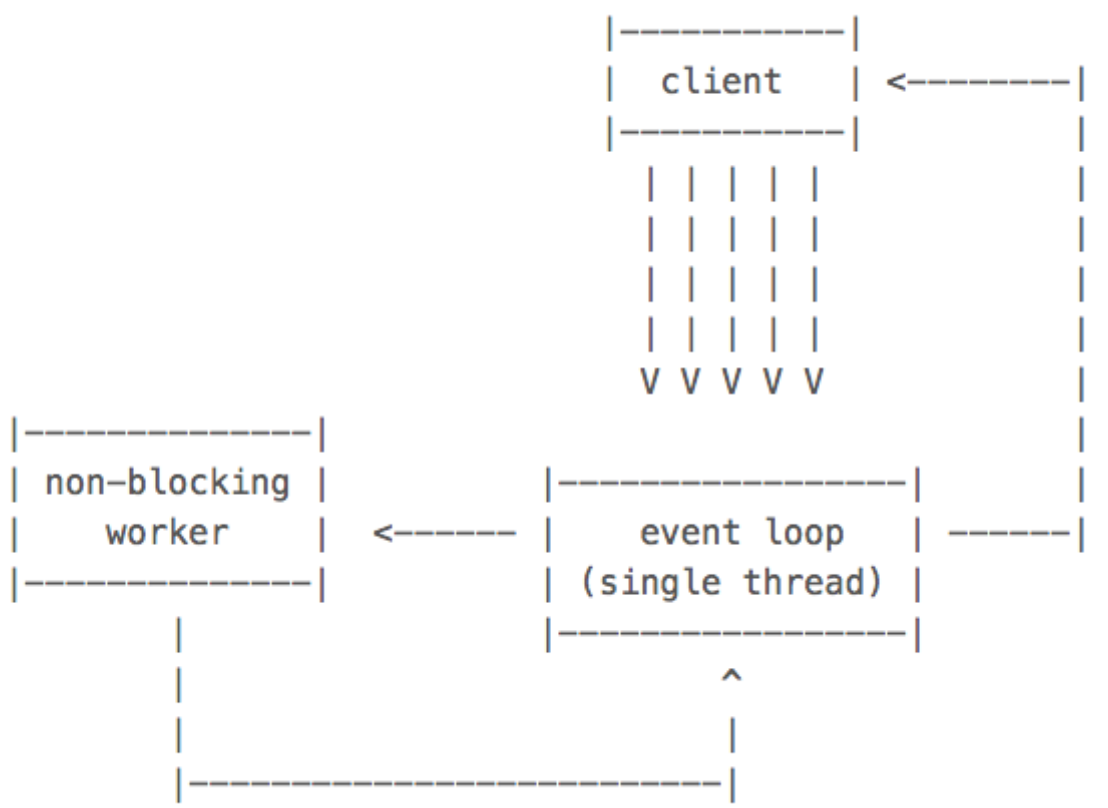
- a key advantage touted for Node.js is its speed
- many companies have noted the performance benefits of implementing Node.js
  - *including PayPal, Walmart, LinkedIn...*
- a primary reason for this speed boost is the underlying architecture of Node.js
- Node.js uses an **event-based** architecture
- instead of a threading model popular in compiled languages
- Node.js uses a single event thread by default
- all I/O is asynchronous

Server-side considerations - Node.js

intro - conceptual model for processing in Node.js

- how does Node.js, and its underlying processing model, actually work?
- client sends a hypertext transfer protocol, HTTP, request
  - *request or requests sent to Node.js server*
- event loop is then informed by the host OS
  - *passes applicable request and response objects as JavaScript closures*
  - *passed to associated worker functions with callbacks*
- long running jobs continue to run on various assigned worker threads
- responses are sent from the non-blocking workers back to the main event loop
  - *returned via a callback*
- event loop returns any results back to the client
  - *effectively when they're ready*

Image - Client-side and server-side computing



Node.js - conceptual model for processing

Server-side considerations - Node.js

intro - threaded architecture

- concurrency allows multiple things to happen at the same time
- common practice on servers due to the nature of multiple user queries
- Java, for example, will create a new thread on each connection
  - *threading is inherently resource expensive*
- size of a thread is normally around 4MB of memory
- naturally limits the number of threads that can run at the same time
- also inherently more complicated to develop platforms that are thread-safe
  - *thereby allowing for such functionality*
- due to this complexity
  - *many languages, eg: Ruby, Python, and PHP, do not have threads that allow for real concurrency*
  - *without custom binaries*
- JavaScript is similarly single-threaded
  - *able to run multiple code paths in parallel due to* ***events***

Server-side considerations - Node.js

intro - event-driven architecture

- JavaScript originally designed to work within the confines of the web browser
- had to handle restrictive nature of a single thread and single process for the whole page
- synchronous blocking in code would lock up a web page from all actions
  - *JavaScript was built with this in mind*
- due to this style of I/O handling
  - *Node.js is able to handle millions of concurrent requests on a single process*
- added, using libraries, to many other existing languages
  - *Akka for Java*
  - *EventMachine for Ruby*
  - *Twisted for Python*
  - ...
- JavaScript syntax already assumes events through its use of callbacks
- **NB:** if a query etc is CPU intensive instead of I/O intensive
  - *thread will be tied up*
  - *everything will be blocked as it waits for it to finish*

Server-side considerations - Node.js

intro - callbacks

- in most languages
  - *send an I/O query & wait until result is returned*
  - *wait before you can continue your code procedure*
- for example, submit a query to a database for a user ID
  - *server will pause that thread/process until database returns result for ID query*
- in JS, this concept is rarely implemented as standard
- in JS, more common to pass the I/O call a **callback**
- in JS, this **callback** will need to run when task is completed
  - *eg: find a user ID and then do something, such as output to a HTML element*
- biggest difference in these approaches
  - *whilst the database is fetching the user ID query*
  - *thread is free to do whatever else might be useful*
  - *eg: accept another web request, listen to a different event...*
- this is one of the reasons that Node.js returns good benchmarks and is easily scaled
- **NB:** makes Node.js well suited for I/O heavy and intensive scenarios

Demos

Travel notes app - series 4

- 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 - JS & Libraries

- Flickr API
- Public feeds
- Public feed - public photos & video
- jQuery
- jQuery
- jQuery API
- jQuery - deferred
- jQuery - .getJSON()
- jQuery - JSONP
- jQuery :parent selector
- jQuery - promise
- MDN
- MDN - JS Objects
- Node.js
- Node.js home
- Node.js - download
- Various
- Create your own AJAX loader
- W3
- W3 - JS Object
- W3 - JS Performance