Human-Computer Interaction Design

COGS120/CSE170 - "Intro. HCI"

Instructor: Philip Guo

Lab 5 - Integrating frontend and backend (2016-10-27)

by Michael Bernstein, Scott Klemmer, and Philip Guo

This lab connects what you've learned in Labs 1 to 4

Git
HTML
CSS
JavaScript
Node.js

Today we put together Labs 1 to 4 …

Portfolio website

Browser

Server-side

Frontend development

- HTML for page structure: Lab 1
- CSS for styling: Lab 2
- JavaScript for interaction: Lab 3
- Ajax for updates w/out reload: Lab 6

Backend development

- Express server-side web framework for Node.js: Lab 4
- Data storage (future)

This entire lab is a practice exercise: You will build a dynamic web site

You will need this knowledge to complete assignments A5 and beyond, so make sure you understand this lab!

We're creating a social network site for HCI friends.

My HCI Friends

Doug Engelbart

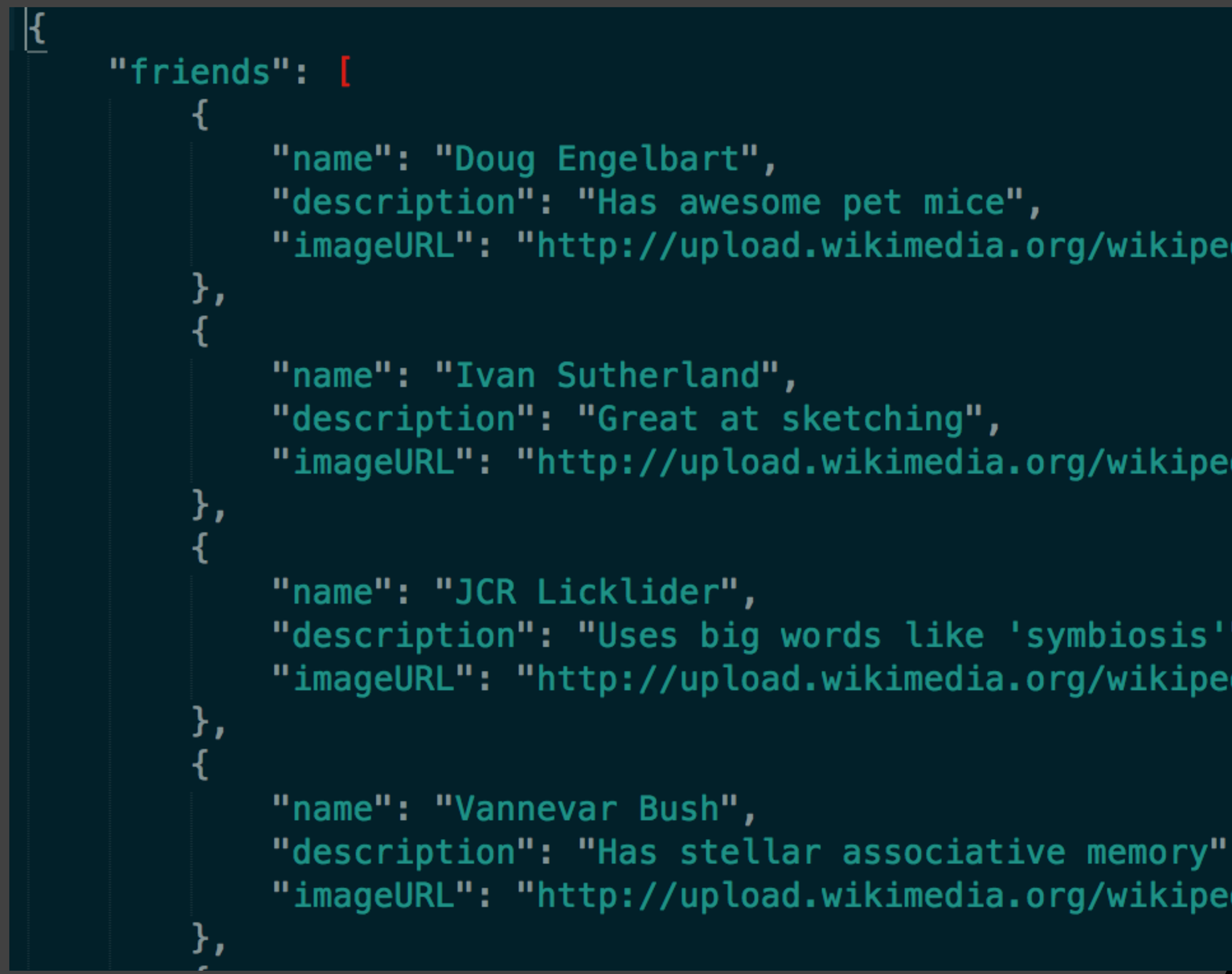
Has awesome pet mice

Get the starter code (the same fork/clone drill)

- Fork the repository: https://github.com/pgbovine/lab5
- `git clone` your forked repository into your introHCI directory

Open lab5/data.json in your text editor

- We are linking this JSON data file to your controllers
- You can write to the object and change it, but it will revert when you restart your Node.js server, since the server loads from data.json whenever it starts up.



Open lab5/routes/index.js

```
// Get all of our friend data
var data = require('../data.json');

exports.view = function(req, res){
    console.log(data);
    res.render('index');
};
```

- We included a line at the top for you:
  `var data = require('../data.json');`
- This loads our JSON fake "database" file into the `data` variable. This will get printed to the terminal console via `console.log` whenever you load the page.

Start node.js

- `cd lab5` to enter the lab 5 directory from introHCI
- To start node: `node app.js`

Verify that the basics work

Visit this URL: http://localhost:3000

Website in browser

Terminal console output

```
{ friends:
   [ { name: 'Doug Engelbart',
       description: 'Has awesome pet mic
       imageURL: 'http://upload.wikimedi
8.jpg/972px-Douglas_Engelbart_in_2008.jp
     { name: 'Ivan Sutherland',
       description: 'Great at sketching'
       imageURL: 'http://upload.wikimedi
     { name: 'JCR Licklider',
```

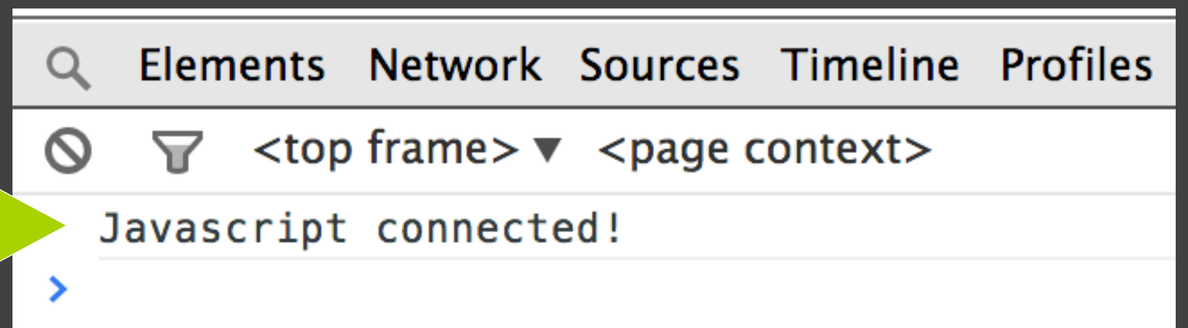
Remember: to see your code changes, restart Node.js in your terminal and reload the page in your web browser
(yep, it's tedious, we know)

Act One:

Connect JavaScript and CSS

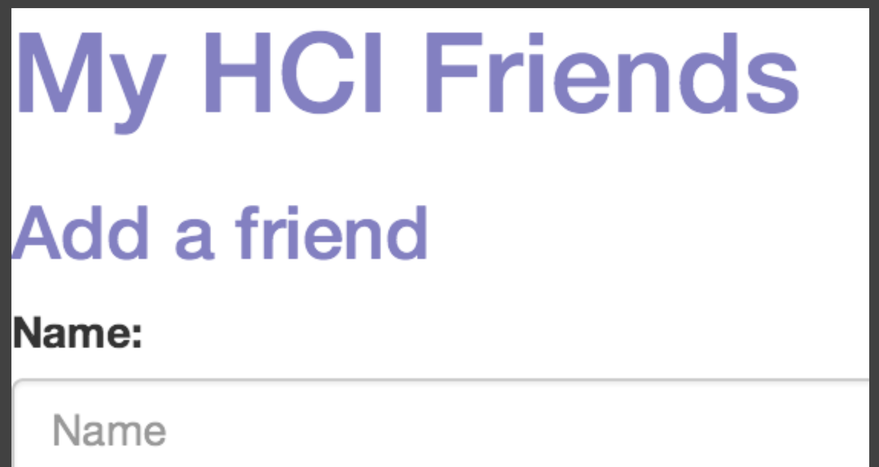
Import JavaScript into your HTML

- *What do I do?*
  - Import the JavaScript files into the HTML file (where is that file?)
- *How do I do it?*
  - Hint: use `<script>` tags
  - The HTML file you want lives in the views folder
  - The Javascript files are in the public/js/ directory: jquery-1.11.0.js, bootstrap.js, hci-friends.js. Import them in that order, because bootstrap and hci-friends depend on jQuery. Think about what their exact filenames should be in the `<script>` tags. *(not intuitive!)*
- *How do I know if it's working?*
  - "Javascript connected!"



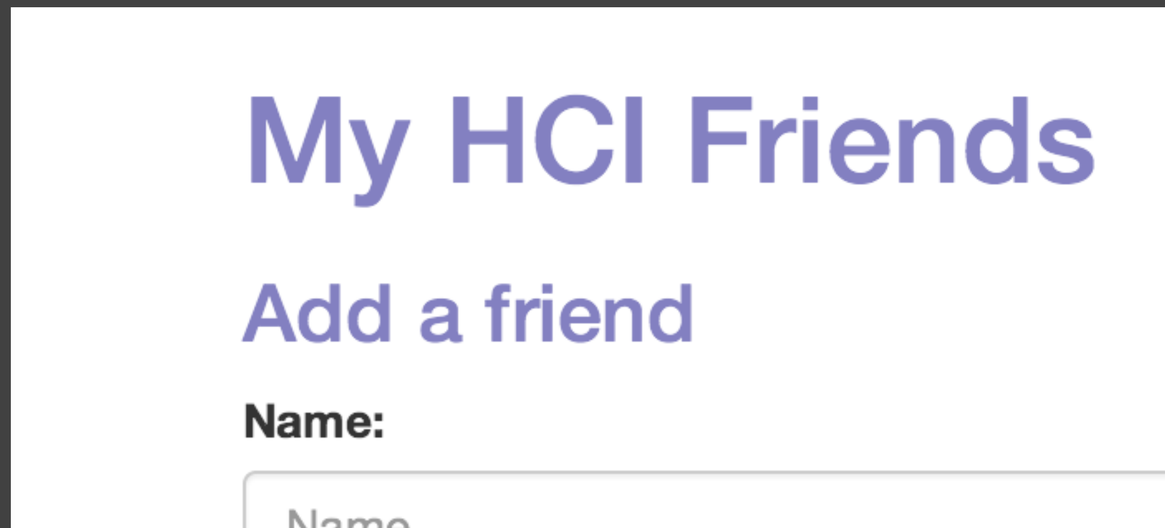
Import CSS into your HTML

- *What do I do?*
  - Import the CSS files into the HTML file
- *How do I do it?*
  - Use `<link>` tags
  - CSS lives in the public/css/ directory: bootstrap.css, bootstrap-theme.css, hci-friends.css. Import them in that order, because hci-friends overwrites styles from the other two. Again, think carefully about what their filenames should be.
- *How do I know if it's working?*
  - Check the fonts (Bootstrap) and colors (hci-friends)



Set up Bootstrap container for page content

- *What do I do?*
  - Create the `<div class="container">…</div>` in the HTML file to surround the entire page's content, which will have Bootstrap automatically create margins
- *What is a Bootstrap container?*
  - It is a div that your page content should go in
  - How do I know if it's working?
  - Look for page margins on left & right

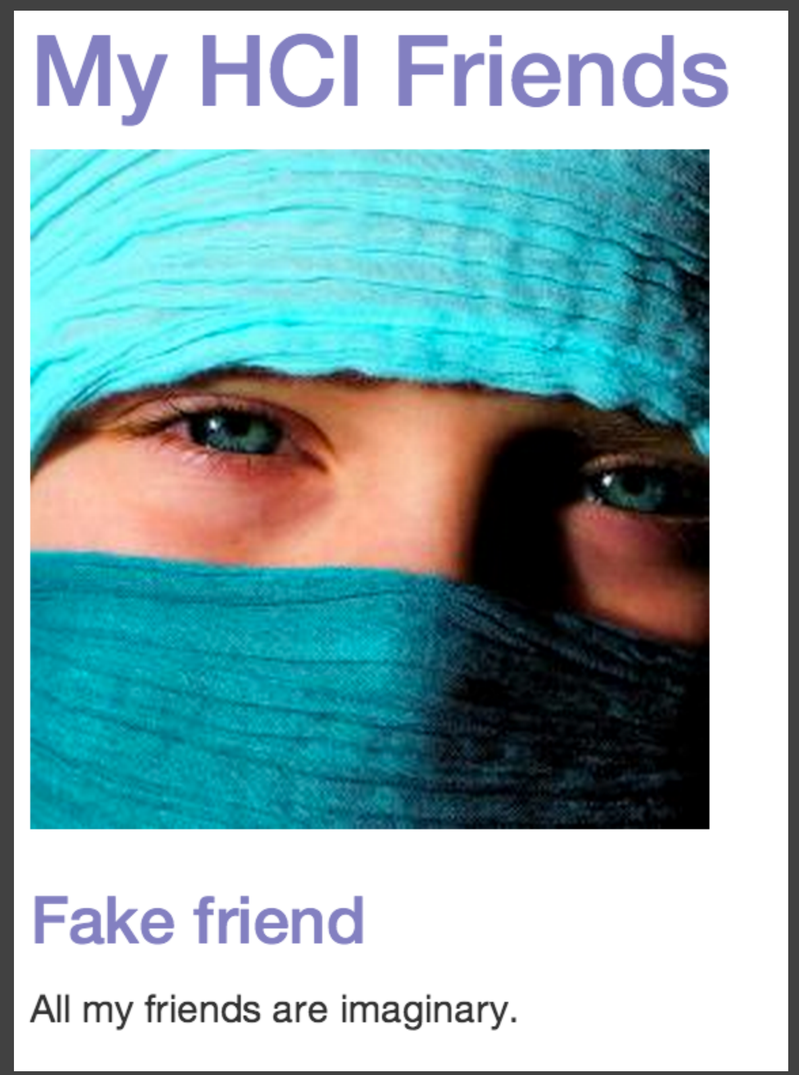


Act Two:

Render the friends

Stub out the HTML for one friend

- *What do I do?*
  - Hand code one friend div directly into the index web page, above the "Add a friend" form. Don't use template variables yet.
  - It should look like the example to the right: image, name, and description.
  - A friend should have…
    - Image: http://lorempixel.com/500/500/people
    - Name: use a smaller header element, like h3
    - Description text: normal paragraph text
  - We recommend placing the friend elements inside a parent div.



Pass the friend JSON to the template

- *What do I do?*
  - Get the `data` variable from the `require()` statement in index.js, and send it as an argument to `res.render()` in the index.js controller.
  - You can pass `data` along directly, no need to wrap or change it. Its contents will be accessible from within the template.
  - Don't forget to restart node.js when you make changes!
- *How do I know if it's working?*
  - Right now, there will be no feedback. We'll use it in a moment.

Render all the friends into the page

- *What do I do?*
  - Edit the template so it places the data you just passed it into the HTML. Use the Handlebars language to access it.
  - Write all the friends using an `each` block. Make sure you have checked the property name in the JSON. How should you access it?
  - Remember to restart node.js when you make changes.
- *Common errors*
  - *The friends are squished on top of each other. It looks bad.* Try adding an `<hr>` horizontal rule at the bottom of every friend div
  - *Nothing's showing up.* Try adding a debug property to the data and see if you can access that, rather than the "friends" property. If you don't access the correct field, it will write out an empty string.

Reign in the images

- *What do I do?*
  - Step 1) Write a CSS rule in hci-friends.css that sets the width for the friend divs. It should set the maximum width for any friend. Scope the CSS rule so that it applies to the entire friend div rather than just the `img`.
  - Step 2) Add the Bootstrap class `img-responsive` to the images. Otherwise they won't respect the friend div's width.
  - For fun, Bootstrap's `img-rounded` adds rounded corners.
- *Common errors*
  - *I set the width on the containing div but the images are still big.* Did you set the `img-responsive` class on the images themselves?

Our current state

My HCI Friends

Doug Engelbart

Has awesome pet mice

Act Three:

Add interactivity

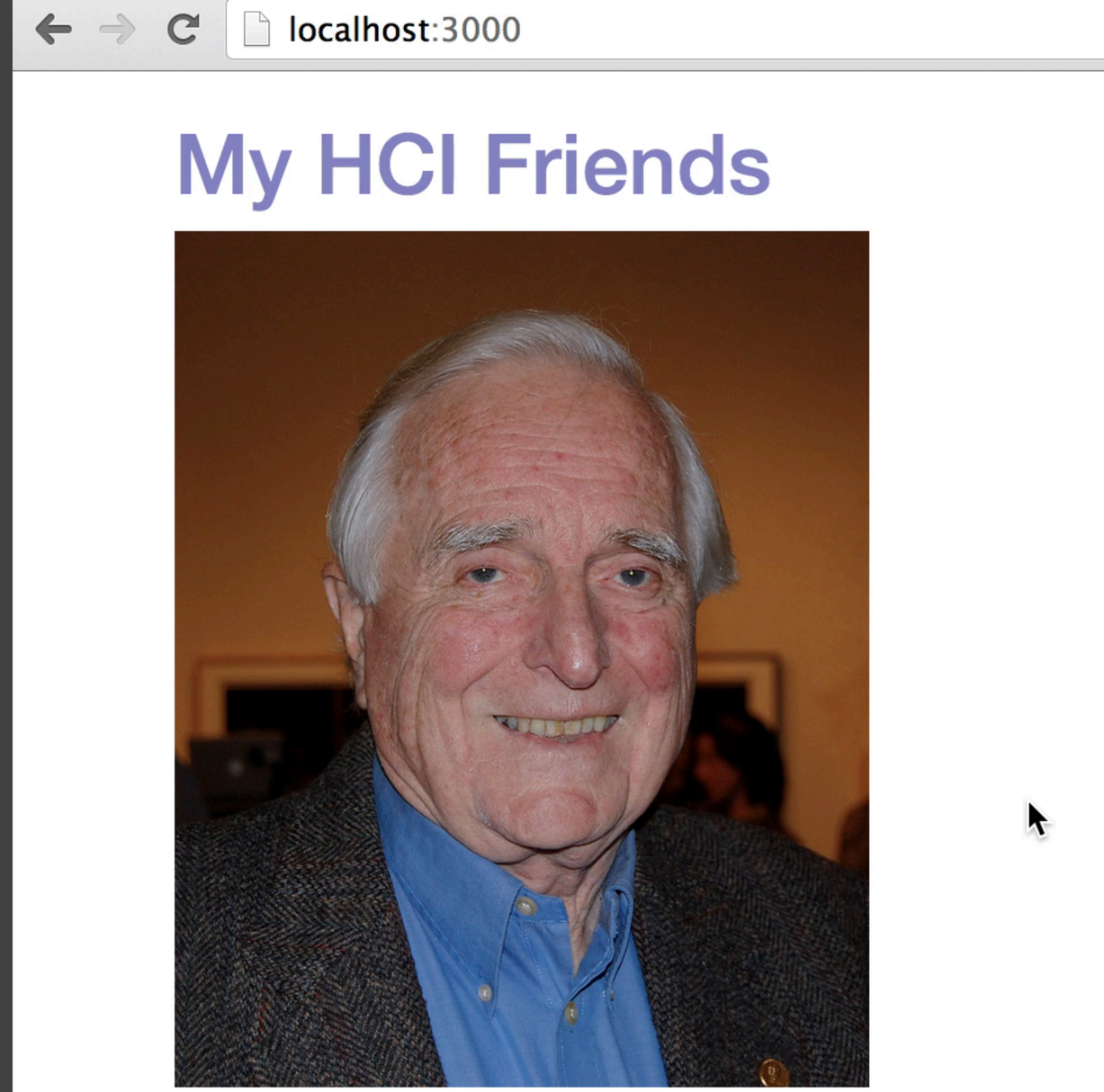
Listen to clicks on the friend names

- *What do I do?*
  - Add a click listener on the friend name.
  - Though it's possible to add a click listener to any element, the affordances are better if you use an `<a>` element. Wrap the name in an `<a>` element and set the href to `href="#"`
  - Then, select the names using jQuery in hci-friends.js and register a click listener. Adding a distinctive class to the friend divs makes selection easier.
- *I'm stuck!*
  - Think about the selector `$("selector")` you need to register the click listener. Add extra CSS classes if it helps.

Replace the name with an anagram

- *What do I do?*
  - Write Javascript so that, when the name is clicked, you replace the name with an anagrammed version.
  - Make sure to call `preventDefault()` in the click listener to that the page doesn't try to follow the link and reload (default behavior)
  - Use `$(this).text()` to retrieve the current name from inside of the click listener function
  - Use `anagrammedName(name)` in hci-friends.js to return an anagrammed version of the name
  - See Lab 3 for details. Important key functions are `$(this)` and `text(newtext)`

Our current state (which anagrams to “Ornate structure”)

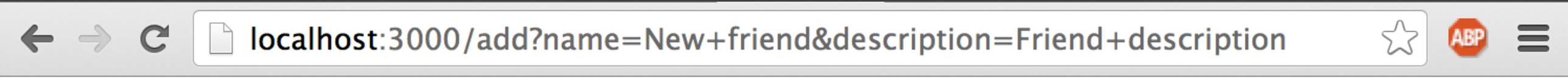


Act Four:

Make new friends

Try submitting the "Add friend" form now

- You should see the URL it wants to submit:



- Error! There is no handler for the /add URL. So, you'll need to add a new controller to handle requests for the /add URL

Register the route

- *What do I do?*
  - Open app.js and add a `require()` statement near the line
    `var index = require('./routes/index')`
    to load a new route that is called `./routes/add`
  - For consistency, store the result into `var add = require( ____ )`
  - We'll add the functionality into `routes/add.js` in a moment
- *How do I know if it's working?*
  - Keep going, there's one more slide…

Register the route (2)

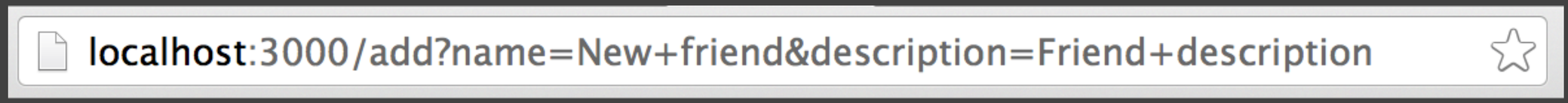
- *What do I do?*
  - Register the URL localhost:3000/add by including an `app.get()` statement near the line
    `app.get('/', index.view)`
    that registers a new route to `/add` via the function `add.addFriend`
  - Don't worry yet about trying to capture the form's variables
- *How do I know if it's working?*
  - Restart node.js. The node.js console should tell you that "yay, addFriend just ran!"
  - (Before, the node.js console would say code 404 for this URL: e.g., there is no controller registered for that URL.)

Create the controller

- *What do I do?*
  - Open add.js inside the routes folder:
    `exports.addFriend = function(req, res) { ... }`
  - Add your new friend's data into the `data` object
    - see the next few slides for more details
  - Render index.handlebars in the `addFriend` function
- *How do I know if it's working?*
  - Submitting the form should render the index page but with your new friend listed on there (see the next few slides for details)

Construct the fake friend's JSON object

- Forms by default send information in this format:



- *What should I do?*
  - Go access these variables in the controller, using `req.query.name` and `req.query.description`
  - Create a new JSON object for the friend, following the format in data.json. Use a fake imageURL property: `http://lorempixel.com/400/400/people`
  - `console.log()` the object to make sure it looks correct
- *How do I know if it's working?*
  - The node.js console should output the object you want to add

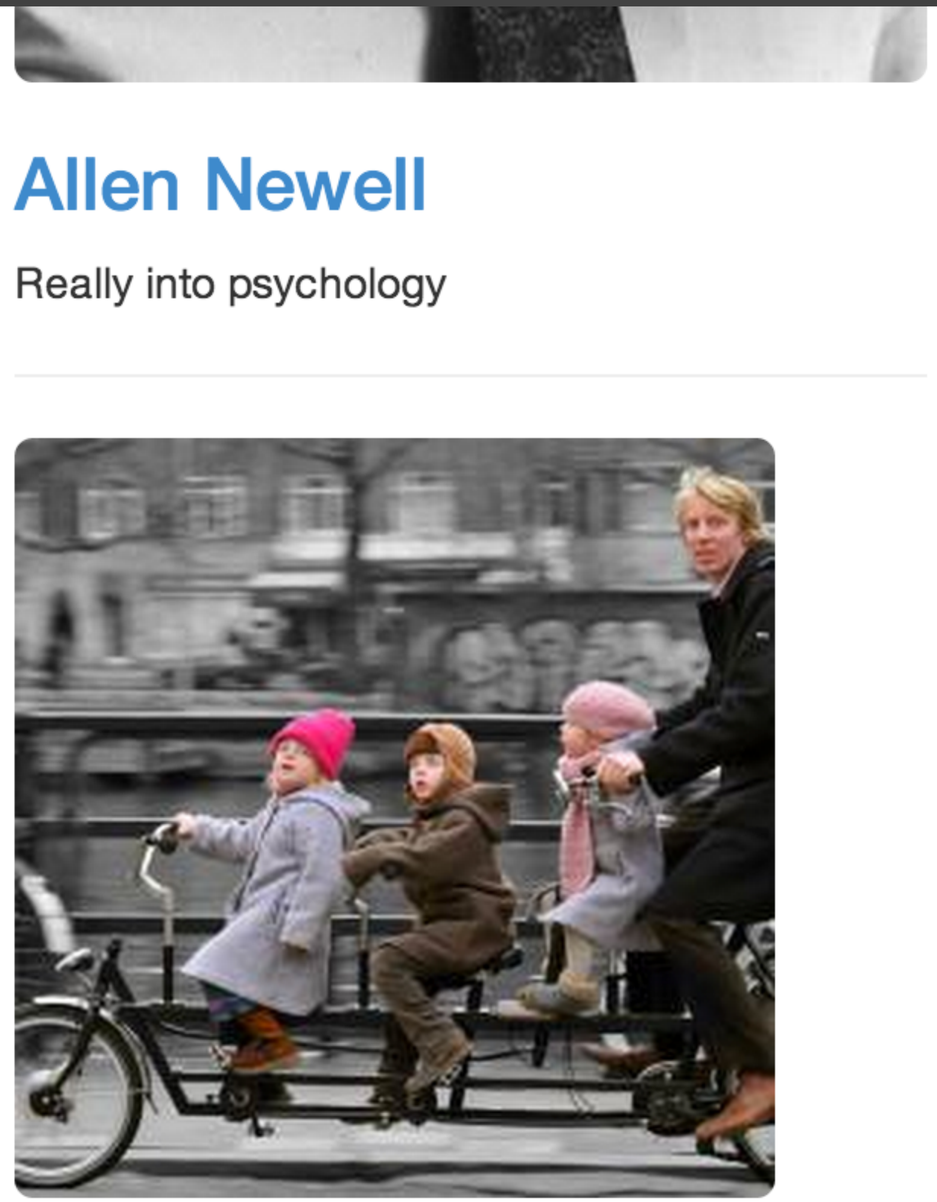
Add the friend to the fake "database"

- *What should I do?*
  - To add a new friend to the array of friends use Javascript's `push()` function for arrays. For example:

    `data.friends.push(newFriend)`
  - The shared data is accessible via the `data` variable, since we imported it using `require()` at the top of the file
- *How do I know if it's working?*
  - Check the next slide

Verify that the new friend appears on the home page

- `push()` adds to the end of an array, so the friend will be at the bottom of the page
- Since this is Javascript data, it only remains until you Ctrl-C in the console to stop node.js



Stretch goal:
Add additional social network-style functionality to the site.

e.g., news feed

e.g., profile pages for friends