Assignment 0 Recap
- Store is **persistent**
  - Yes, your previously stored data will remain in your app
- **Not** the same as **temporary** JSON data that was used in COGS 120
- Persistency testing - close the server and rerun the server
  - Is the data still there?
  - If so, then it’s most likely persistent
Routing
GET
1. User goes to address, requests a resource

GET request to “/”
2. Server handles the request

```javascript
var router = {
    index: require('./routes/index'),
    message: require('./routes/message')
};

// Routes
app.get('/', router.index.view);
app.post('/message', router.message.send);
```
3. Route renders view

```javascript
var models = require('..\models');

exports.view = function (req, res) {
    models.Message.find().exec(renderMessage);

    function renderMessage(err, message) {
        if (err) console.log(err); // naive error handling
        res.render('index', { 'data': message });
    }
};
```
4. Rendered!

Email: test@test.com
Content: test

My Messages

<table>
<thead>
<tr>
<th>Email</th>
<th>Content</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:test@test.com">test@test.com</a></td>
<td>test</td>
<td>Wed Dec 31 1969 16:00:00 GMT-0800 (PST)</td>
</tr>
</tbody>
</table>
POST
1. User submits to the form

Email: test@test.com
Content: test

My Messages

<table>
<thead>
<tr>
<th>Email</th>
<th>Content</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:test@test.com">test@test.com</a></td>
<td>test</td>
<td>Wed Dec 31 1969 16:00:00 GMT-0800 (PST)</td>
</tr>
</tbody>
</table>

POST request to “/message”
2. Form submission makes a request

```html
<form action="/message" method="post">
  Email: <input type="text" name="email"><br>
  Content: <textarea name="content"></textarea><br>
  <input type="submit" value="Submit">
</form>
```
3. Server handles this request

```javascript
var router = {
    index: require('./routes/index'),
    message: require('./routes/message')
};

// Routes
app.get("/", router.index.view);
app.post('/message', router.message.send);
```
4. Route executes and redirects

```javascript
var models = require('./models');

exports.send = function (req, res) {
    console.log(req.body);
    var newMessage;
    var date = new Date();
    if (req.body.email && req.body.content) {
        newMessage = new models.Message({
            'email': req.body.email,
            'content': req.body.content,
            'created': date.getDate()
        });
        newMessage.save(function (err) {
            if (err) console.log(err); // naive error handling
            res.redirect('/');
        });
    }
};
```
5. Message Added!

My Messages

<table>
<thead>
<tr>
<th>Email</th>
<th>Content</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:test@test.com">test@test.com</a></td>
<td>test</td>
<td>Wed Dec 31 1969 16:00:00 GMT-0800 (PST)</td>
</tr>
<tr>
<td><a href="mailto:test@test.com">test@test.com</a></td>
<td>test</td>
<td>Wed Dec 31 1969 16:00:00 GMT-0800 (PST)</td>
</tr>
</tbody>
</table>
Assignment 1 Intro
Assignment 1 Structure

- Part I
  - Introduces Passport.js and Socket.io
  - Similar structure as Assignment 0; follow steps to create an app
- Part II
  - Create your own social media application
  - Integrate knowledges from Part I
- Both parts should be done in your assigned group
Part I

- **Passport.js**
  - Authentication tool for Node.js
  - Helps developer build features involving login, sign-up, and profiles
  - Capable of creating custom login or using third-party source (e.g. Twitter)
  - [http://passportjs.org/](http://passportjs.org/)

- **Socket.io**
  - Simple websocket tool for Node.js
  - Useful for features that require live messaging and notification
  - Commonly used for chat application
  - [http://socket.io/](http://socket.io/)
Part I Demo
Part II

- Create your own social media application
- Choose a theme
  - E.g. an application for rock climbers to share experiences of and ratings for climbing routes
  - Tip: try to choose a theme that is specific and simple as possible
- Utilize technologies introduced in Part I
  - Passport.js is required and Socket.io is optional
- Apply usability concepts from Tuesday’s lecture
Grading Criteria

- 2 pts - Assignment 0
- 10 pts - Usability Heuristics (+ README)
- 1 pt - Theme & Target Audience
- 2 pts - Creative & Innovative
- 1 pt - Group Contribution (+ README)
- 2 pts - Peer Evaluation (+ CATME)
- 2 pts - Passport.js
- 1 pt [bonus] - Socket.io
Before Studio

- Read through Assignment 1 wiki
- **Please watch** videos listed under “Before Starting” section
- Bring questions
  - e.g.) how does OAuth and Websocket relate to social media?
- Discuss with your team and your plan for approaching the assignment
Last Tips

- Work co-located as much as possible
  - Teach each other!
- Figure out how to best utilize a 5-strong team
  - Delegation!
- For part 2, pick a theme that interests you, or that you already have knowledge of
  - Would you use it?
- Talk to your studio leader, use office hours, use Piazza
Let’s start now!

Please meet in your assigned Team and start working on the assignments
(TAs will be floating around)