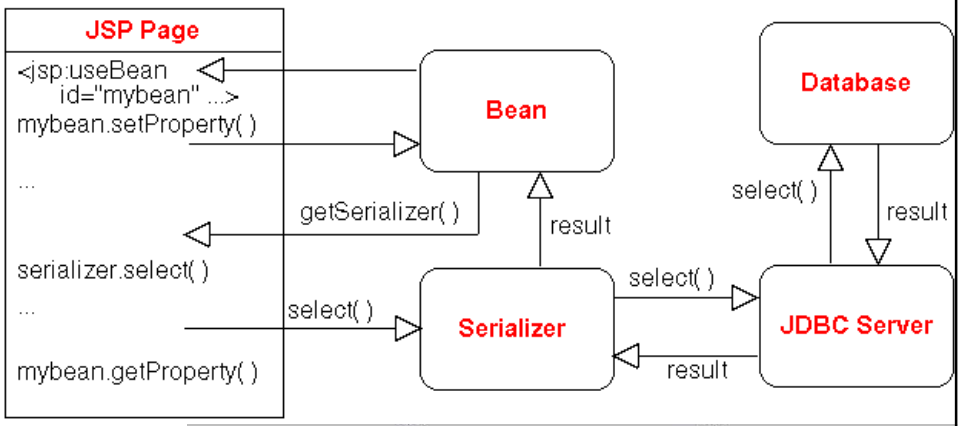
CSCC09F
Programming on the Web

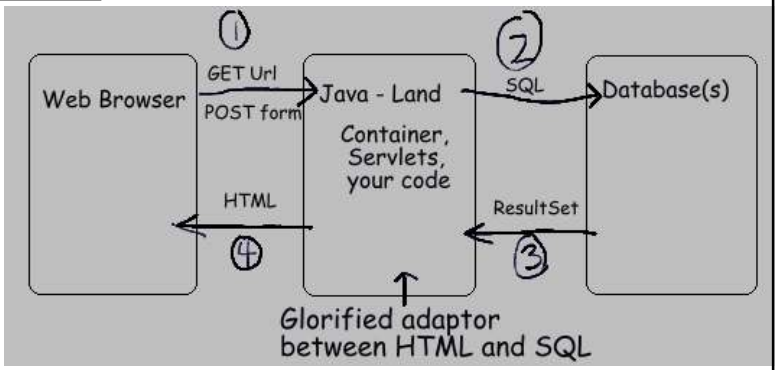
Full Stack JavaScript

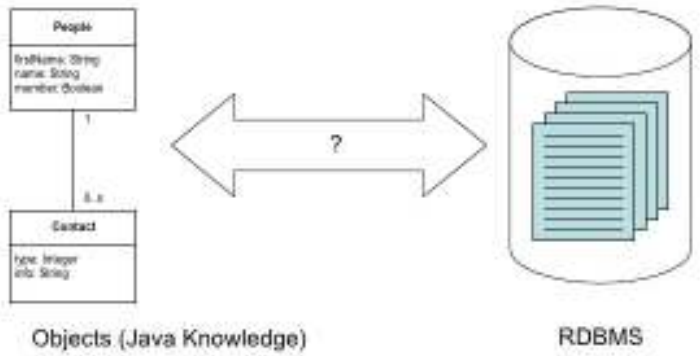
Example for Assignment 2

Classical Apps

- When implementing an app-server using a traditional OO-language such as Java we face a mismatch between client-side JavaScript models and server-side Java objects
- Typically use a server Bean to map client model to server object
- Beans can be populated from forms

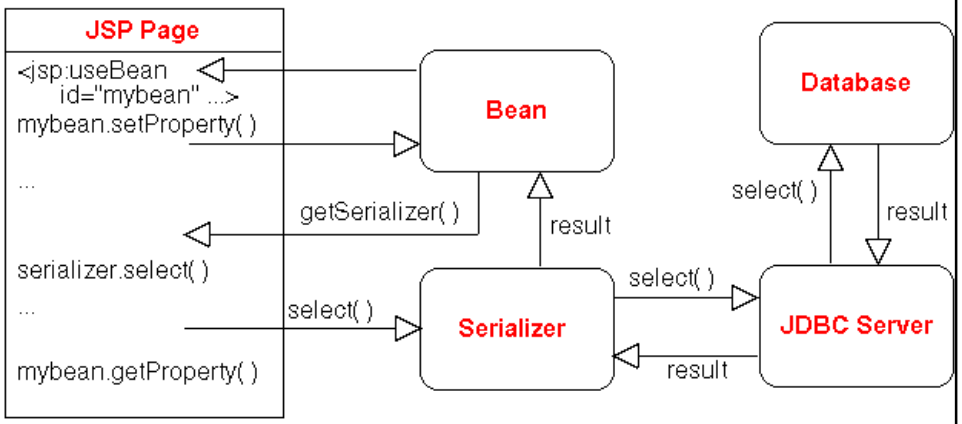


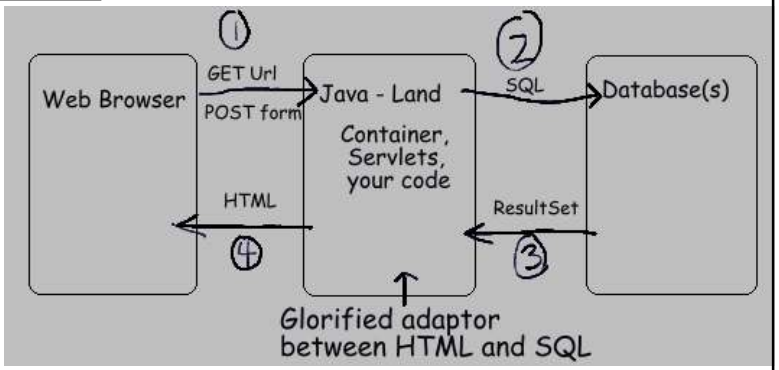


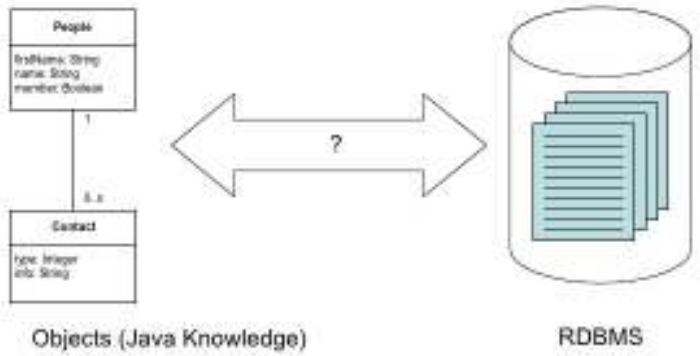


Classical Apps

- When implementing app-server using a language like Java with relational DB we face "object-relational" mismatch
- In the language, domain models represented as objects, in the DB they are represented as rows in tables
- Controller typically has result-set iterator to create beans from DB results

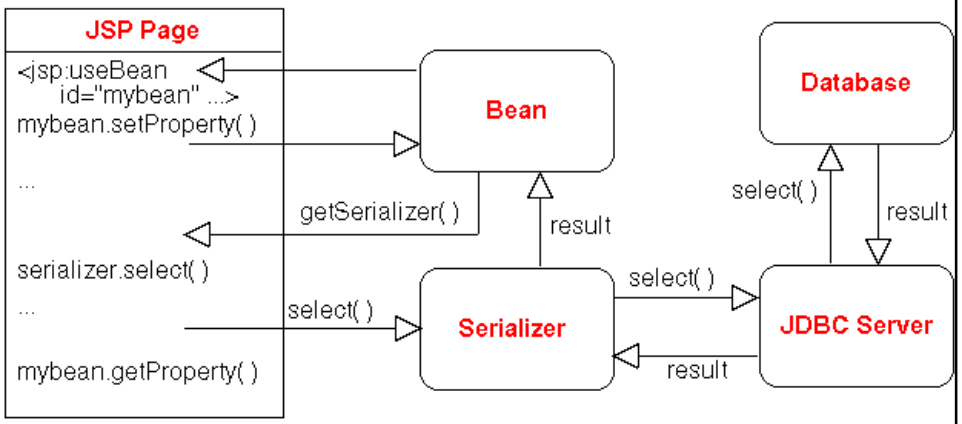


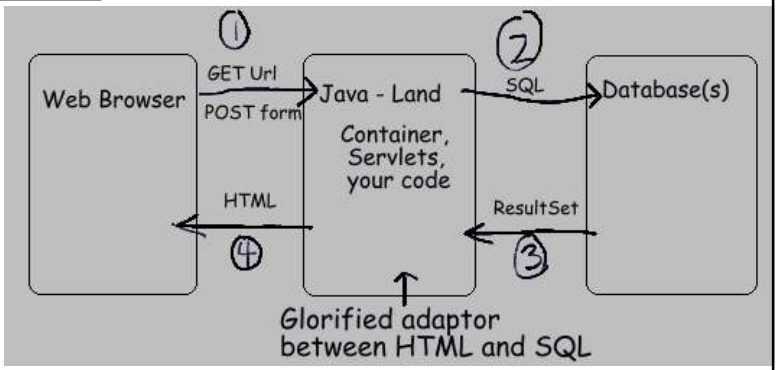


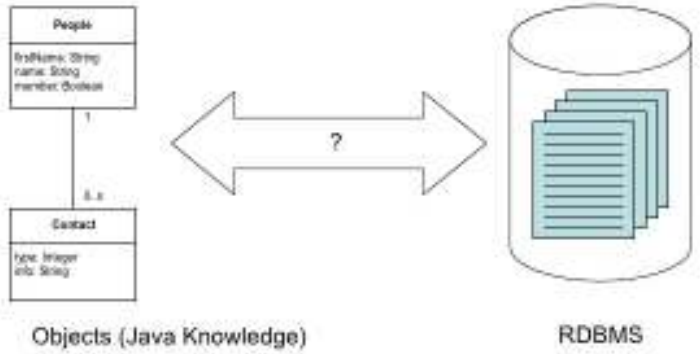


Classical Apps

- When implementing app-server using a language like Java need to map bean results back to client-side objects
- Typically have JSP "view" that renders beans in JSON format
- To get JSON to render on client, use Java tag library to iterate over JSON structure

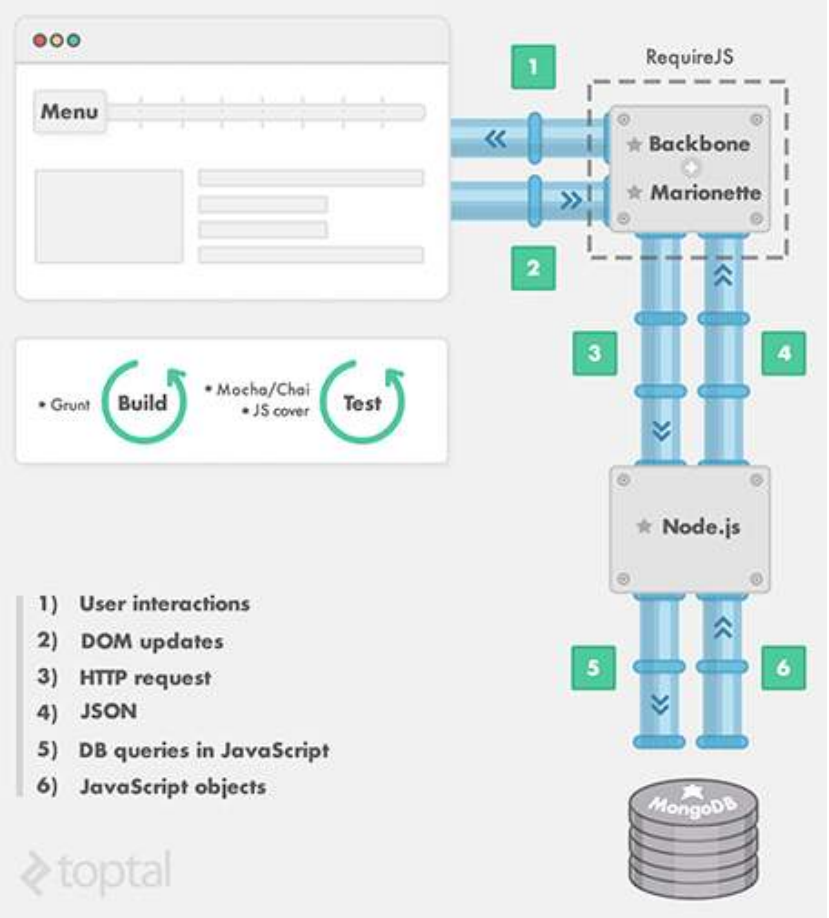






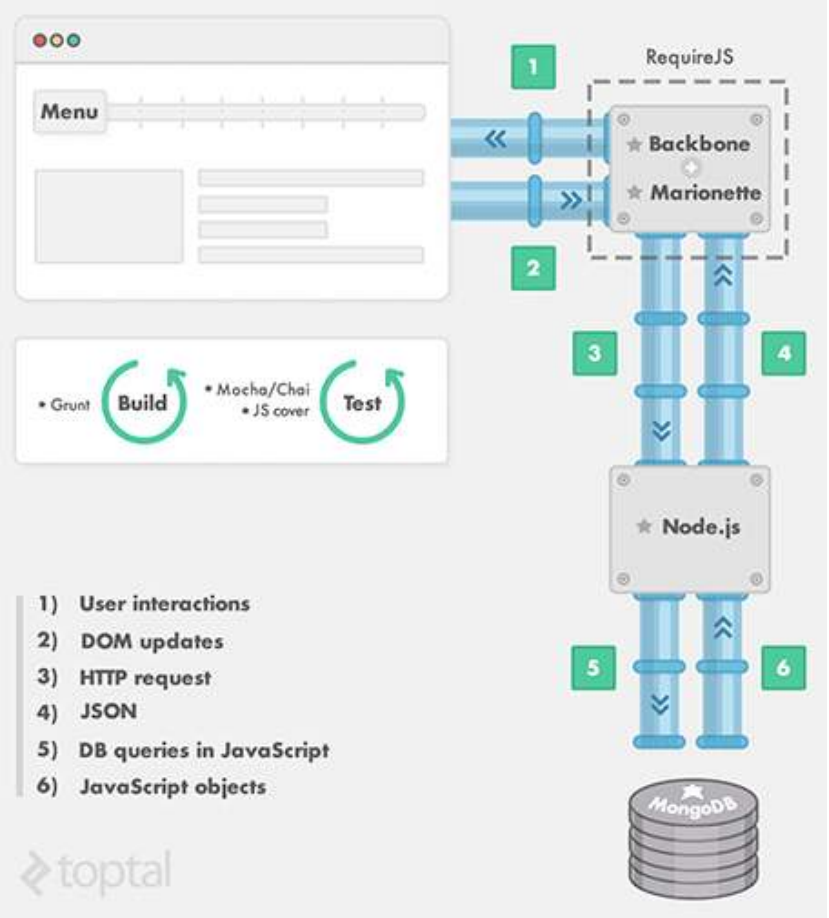
Full-Stack JavaScript

- Full-Stack JavaScript refers to use of JavaScript end-to-end in building an app …
- from the client-side running with a framework like backbone …
- to the server side, running a JavaScript server like Node.js …
- to a database serving up JavaScript data (JSON)



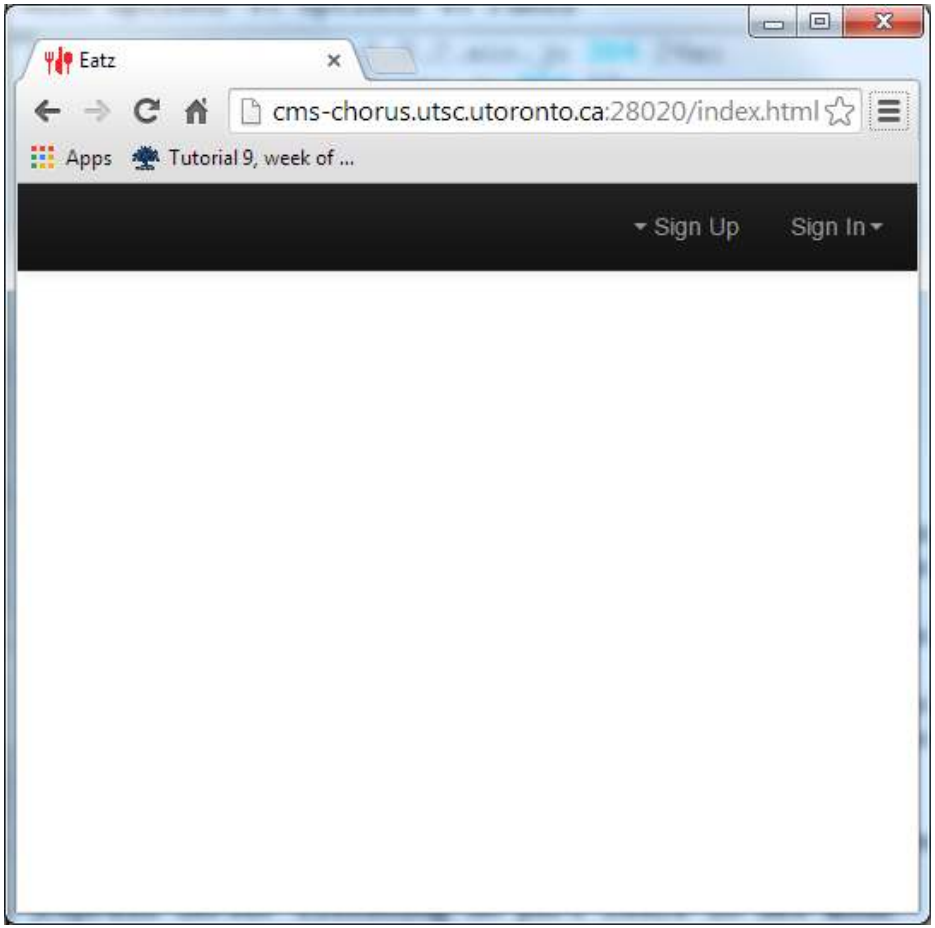
Full-Stack JavaScript

- Agile!
- Take working app … rip out large chunks of code … no complaints from JavaScript (as long as you are careful/consistent in the code left behind)
- Try doing this using a language with strong emphasis on static checking, e.g. Java … forget it!



Full-Stack JavaScript

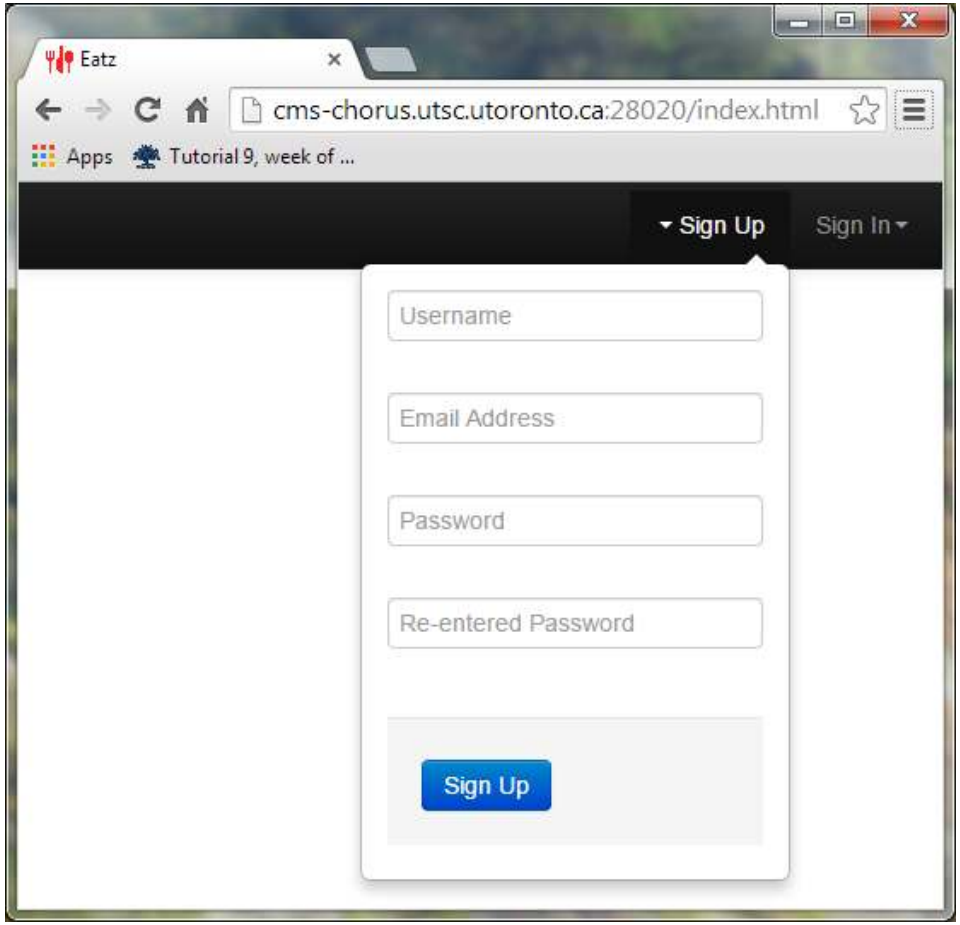
- Implementation of Eatz account-signup
- Most of the used code already posted on lecture/tutorial pages, but this example pulls it all together in working form



Full-Stack JavaScript

Client-side of app:

- static resources
- templates
- router
- models
  - validation rules
- views
  - event handler
  - validation checks
  - Ajax request



Full-Stack JavaScript

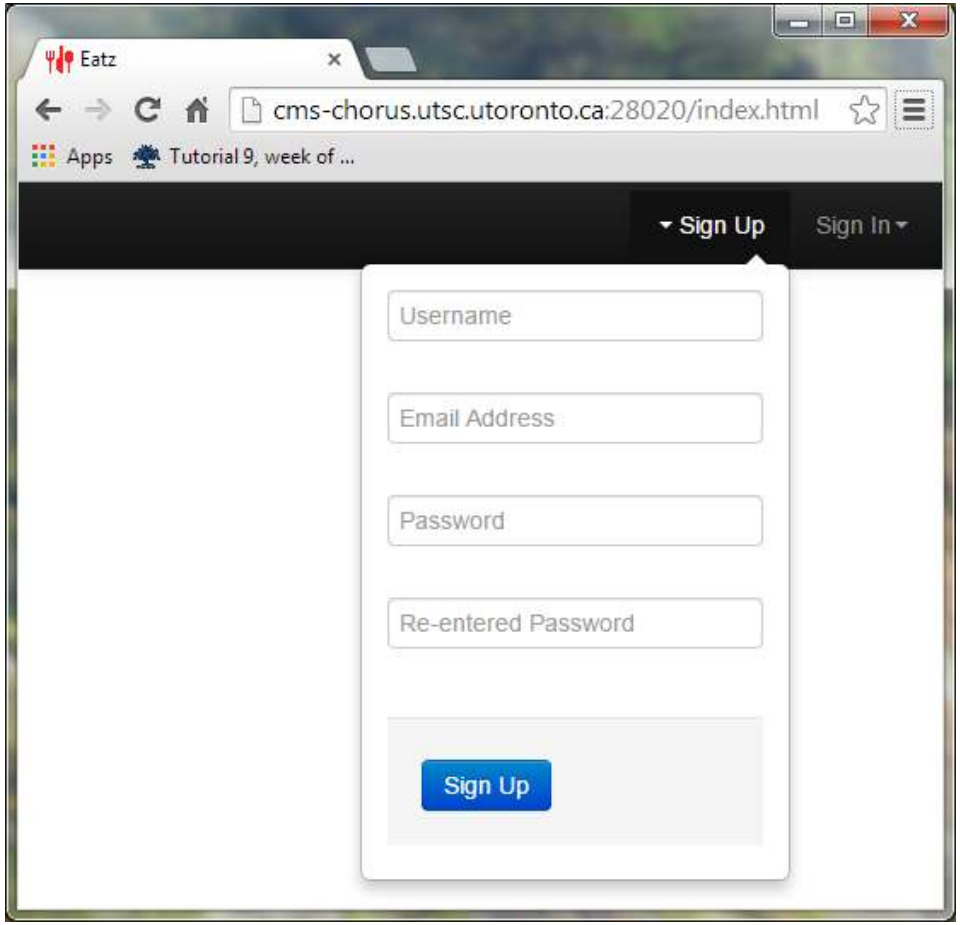
```
<div class="navbar navbar-inverse navbar-fixed-top" >
  <div class="navbar-inner">
    <div class="container">
      <ul class="nav pull-right">
        <li class="dropdown" id="signupdrop">
          <a class="dropdown-toggle" id="register"
                    data-toggle="dropdown">
           <strong class="caret"></strong> Sign Up</a>
          <div id="signup_form" class="dropdown-menu" >
            <!-- Signup form -->
            <form accept-charset="UTF-8">
              <div class="control-group">
                <div class="controls">
                  <input id="signup_username"
                    type="text“ name="username" ...
```

Verbose, but highly regular structure … once you get the pattern it's easy

Full-Stack JavaScript

Server-side of app:

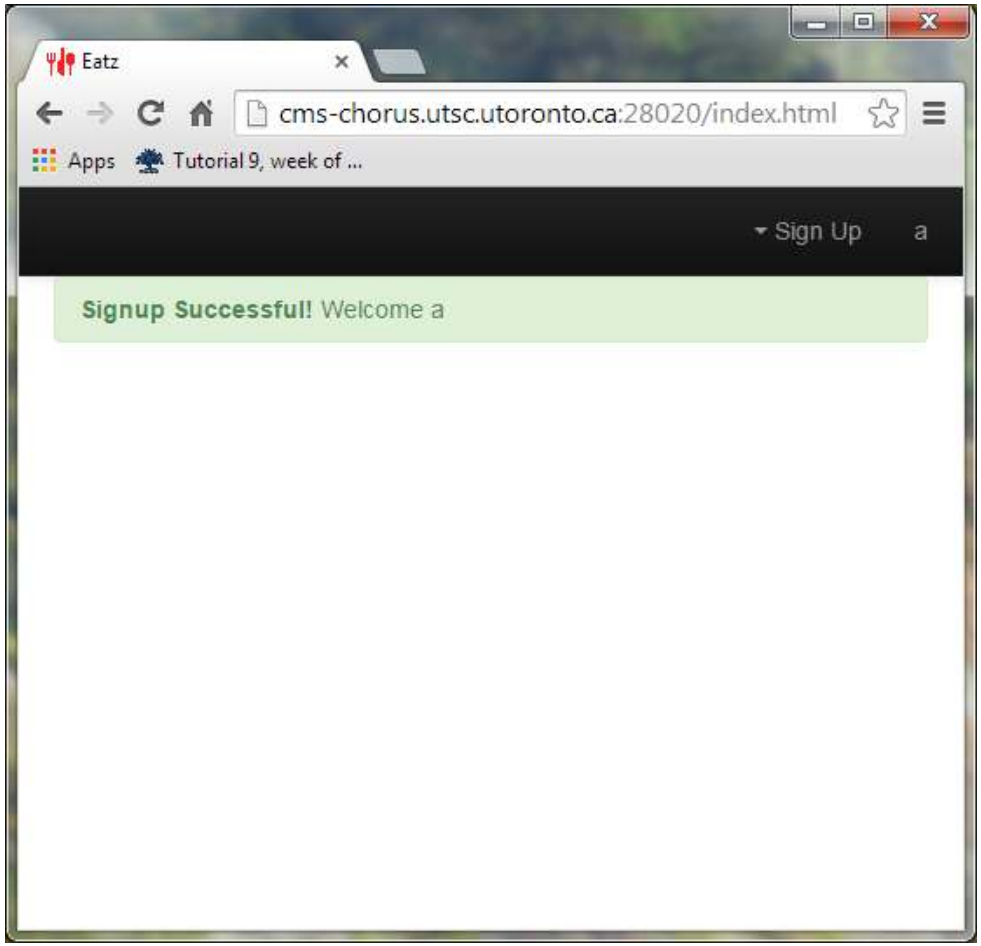
- router
  - middleware setup
  - map routes to handlers
  - start Node server
- route handler
  - request/response processed
- database
  - password set on DB
  - database connection
  - schemas and models
  - collection auto-created



Full-Stack JavaScript

Client side of app

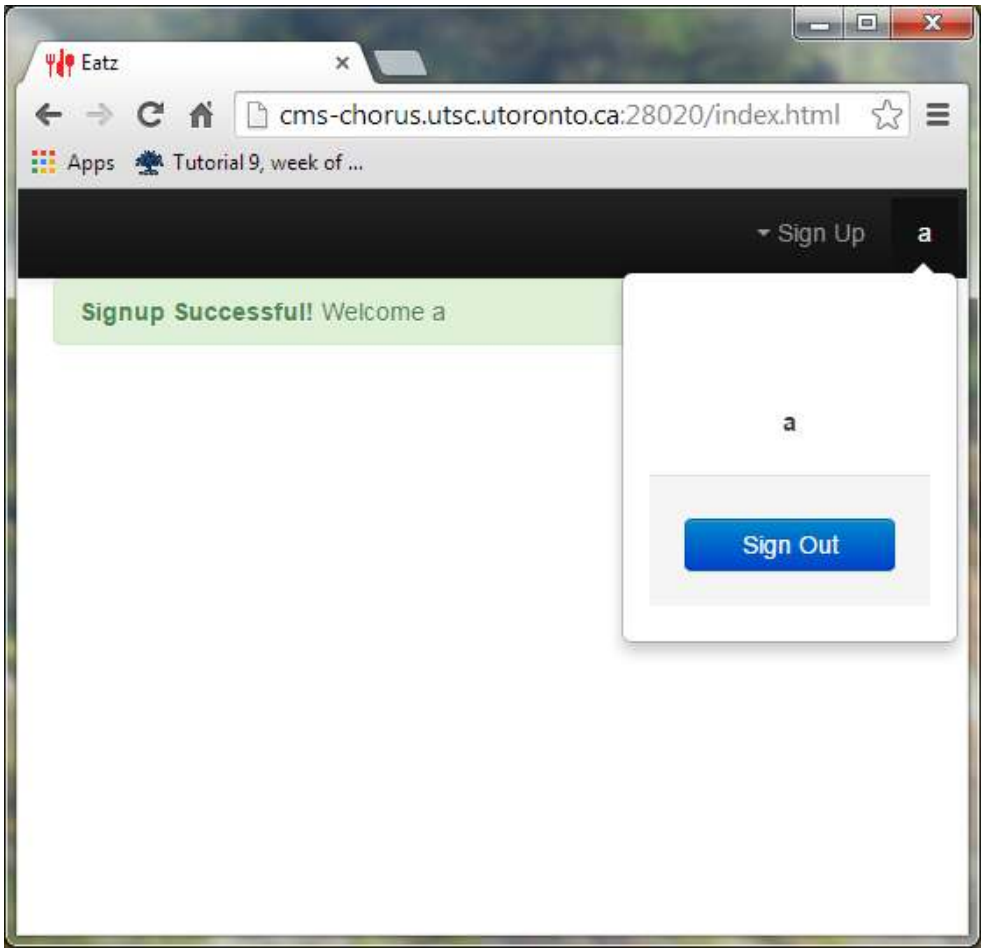
- Ajax response processing



Full-Stack JavaScript

Client side of app

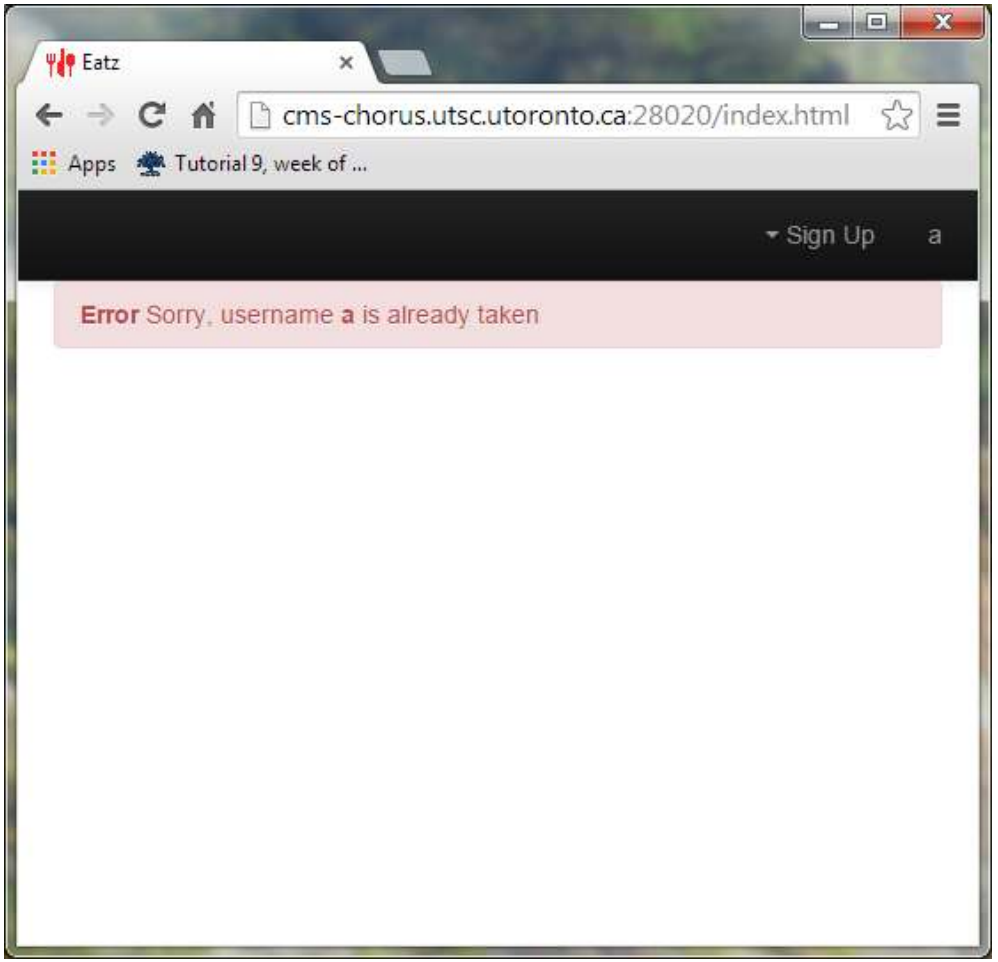
- Ajax response processing



Full-Stack JavaScript

Client side of app

- Ajax response processing



Full-Stack JavaScript

- Example linked to lectures page
- Example directory also posted as tarball