# Day 2: EJS Templates & Web APIs

Re:Coded - Server Side Development Workshop

# Agenda, Review, Housekeeping

**Today we will remove all static placeholder data on our site's frontend with dynamic content from the server.**

1.) Consolidate redundant static markup.

2.) Add dynamic server-rendered content to our markup.

3.) Replace `mocks.js` using jQuery on the client and JSON endpoints on the server.

# Housekeeping

In a group this size, there will be a broad range of backgrounds - and everyone is in their own place on their own journey. I care about each of you as individuals - and am concerned about both going too fast and discouraging you or skipping important concepts - and also of going too slow and not respecting your time. I have a few ideas to help each of us with this...

# A Request...

If it's possible for you, and you're comfortable doing so, it would be a huge help to me, personally, if you could turn on your camera. I'm pretty new to this, and I'm having a hard time gauging how fast to move, whether a topic is interesting or not, etc... We could all use some help when we're learning something new! If you feel comfortable helping me in this way while I learn how to be a more effective trainer, I'd be very appreciative! شكرا!

Along these same lines - I will send you a link to a feedback form after the end of this session. Please fill it out *as honestly as possible*. Your feedback will allow me to adjust things, improve, and make the most of our time together.

# Please join the Slack!

If you have questions that aren't directly related to the workshop, but are still relevant to the group (for example, you'd like to hear more about working for Google vs. Snap vs. Enterprise vs. Consulting vs. Startups, this is a great place for that! I can take the time to answer your question more thoughtfully, and everyone can participate if they choose to (or not).

This also makes it easy for us to chat privately if you need help or have general questions you just don't want to ask publically. At my time at Google and Snap I have interviewed literally hundreds of engineers and ran a few workshops at Google to teach applicants how to put their best foot forward in our interviews. If you have questions about interviewing or getting started in tech - I'm happy to work with you on that as well. If you feel we are going much too slow - reach out to me and I can give you some additional challenges or we can talk about your individual goals and how I can help.

# Scaling Out & Application "Tiers":

I skimmed over this because I felt the background was running too long - but this is a key point that pulls all that together, and I don't want to miss it...

The OSI "Physical Layer" & DNS: I mentioned there were limits to how much we could "scale up" a server - and we talked about how DNS "scales out". We also talked about the OSI "Physical Layer" and the typical "You-Internet-Server" mental model - but we didn't wrap that up by talking about some of the mental models server side engineers use when they think about scaling their applications, why it matters, and talk explicitly about API boundaries. Let's fix that now on the "whiteboard"...

# A Note About Breakout Rooms

Today we will be using the breakout rooms.

Please keep this idea of people's individual journeys in mind today when we go to breakout rooms - this workshop is a safe place for everyone to express their thoughts, ideas, and confusions without judgement or fear of ridicule.  As engineers and scientists we celebrate when we realize we were wrong because it means we have learned something new - that we have grown. The only way this can happen is if we are not afraid to be wrong. It is all of our responsibilities to make sure that is the case for everyone else.

# Questions from Yesterday?

- HTTP (GET/POST, Request/Response)
- Client <--> Server <--> Database
- What is Node.js?
- What is Express?
- What is an Endpoint?
- What is static content and how is it different from dynamic content?
- Start and debug a node.js site using npm & Chrome DevTools

# More Efficient Debugging

nodemon & Package.json scripts

We have a lot of code to write today, so let's fix that whole having to keep starting/stopping node manually...

# Install Nodemon

```
npm install -g nodemon
```

# Add new script in Package.json

```
"scripts": {
  "start": "node ./bin/www",
  "debug": "nodemon --inspect ./bin/www"
},
```

# Now when you want to debug:

```
npm run debug
```

# Consolidating Redundant Markup

## EJS & Express Middleware

Yesterday we looked at the Node.js Hello World program and talked about how we can return any string we'd like, including HTML.

We also learned how to serve static content like HTML files so that we don't need to deal with strings in Node.js.

But what if we want the best of both worlds?

# Why?

- Maintenance
  - Useful software is a living thing

# How?

For each page we will:

- **Pick a path** we actually want for this content our website
- **Add a new endpoint** to handle the request
- **Migrate** the HTML page to EJS templates
- **Implement the new endpoint** by rendering our template with appropriate data

# Example

We will migrate the index.html page together now.

# Individual Exercise

# EJS Cheat-Sheet

`<%= string_variable %>` Include the escaped version of variable in the response

`<%- string_variable %>` Include the string without escaping it in the response

`<% code %>` Code to control program-flow (letting us do cool things like loops and conditionals)

# Let's pick some paths...

GET /posts/recent
GET /posts/trending
GET /posts/create
GET /posts/view

# Express Middleware & Routers

# Express Middleware

Let's look at the docs.

- The order in which we "use" things is the order in which they are given the opportunity to handle a request.
- At each step the middleware can manipulate the request or response.
- And optionally allow middleware processing to continue... or stop.
- You must use a function that calls, or explicitly call response.end yourself.

# Express Middleware

```
app.use(middleware);
app.use(“/path”, middleware);
```

# Express Routers

Let's look at the docs.

- Just a type of middleware.
- Allows you to encapsulate parts of your application.
- When used with the path component of the use function, can be thought of as handling requests for a particular "folder". When used without a path can be thought of as extending the handling of the current "folder".
- Let's look at `index.js` and `users.js` and where these are used in `app.js`, and create a new one for `posts.js`

# Dynamic Server-Rendered Content

## View Models, Data Access APIs

What about the content of our pages?

This seems different from headers and footers which always look the exact same on a given page.

# Why?

- Performance
- Solutions exist to make it easy to "write once and use either way" but are outside the scope of this workshop.

# How?

- Identify each template's data needs (sometimes called a view model) (note; not page).
- Create one function for each template (which retrieves that data).
- Mock the data to start.

# Breakout Exercise

# Group Work:

- List the templates you would create
- List the view models for each template and what data they each contains
- List your data functions

# Let's Review My Solution

Please update your branch tracking origin/master to get my implementation and we will review.

# Dynamic Content via Javascript

jQuery, AJAX

Let's take a look at mocks.js - this is the last part of the web frontend that uses fake data (from the frontend itself).

# AJAX with jQuery

```
$.ajax({
     type: "POST",
     url: “/path/“,
     data: JSON.stringify(obj),
     contentType: "application/json; charset=utf-8",
     dataType: "json",
     success: function(result){
        // Success
     },
     error: function(error) {
        // Non-200 Status
     }
 });
```

# JSON Services with Express

```
router.post('/path', function(req, res, next) {
  var request_body = req.body;

  // …

  res.status(400).send(response);
});
```

# Breakout Exercise

# Group Work:

- List the endpoints you would create & what data you would send and receive.
- What status codes will you return?

# Let's Review My Solution

Please update your branch tracking origin/master to get my implementation and we will review.

# See You Tomorrow!

Please make sure to sign up for the Slack if you want!