HTTP Methods & Endpoint Design

Schedule

1. Learning Objectives
2. Activity: Write an API (45 minutes)
3. BREAK
4. Review Messages API (25 minutes)
5. Activity: Design an API (25 minutes)
6. Wrap-Up
7. Resources

Learning Objectives

By the end of this lesson, students will be able to…

1. Write a RESTful API using Express for one resource.
2. Identify best practices on designing routes, such as middleware and modular routing.
3. Practice pair programming techniques.

Review Homework (20 minutes)

Review the Promises homework and go over any misconceptions.

Activity: Write an API (30 minutes)

Choose a partner who you haven't worked with yet. You'll be working together on today's assignment (but pair programming is

Go to **this tutorial by Robin Wieruch** and follow the steps to create your first API using Node and Express. Make sure to com

As you work, answer the following questions:

- What's npm init doing when you setup your Node.js project?
- What benefit is Nodemon giving us?
- Why do we use Babel?
- Why do we need Environment Variables?
- How do frontend and backend application communicate with each other?
- What is the purpose of separating code into different files for models and routes?
- What do we mean by *modular routing*?

BREAK (10 minutes)

Review Messages API (25 minutes)

Review the concepts in the tutorial as a class.

Middleware

Middleware is a function(s) that are executed before your route is invoked. It can be used for logging requests, determining wh

What are some examples of middleware we used from the tutorial? What made them useful?

What about custom middleware? An example of custom middleware used in the tutorial is:

```
app.use((req, res, next) => {
  req.me = users[1];
  next();
});
```

Custom middleware can be used to execute any code, or make changes to the `req` or `res` objects, before executing the ro

Routers

Express Router is a class which helps us to create route handlers. We can have multiple routers which each handle a subset o `users` router.

```
const messageRouter = express.Router();

messageRouter.get('/', (req, res) => {
    return res.send({'message': 'hello world'});
});

...

app.use('/messages', messageRouter);
```

Typically, we would put each router in a separate file to maintain separation of concerns.

Nested Routes using Modular Routing

Modular routing imposes some structure on our routes - which, if you are building a large website, becomes necessary so that

We can use middleware to extract information sent in nested routes.

Let's say we are building out the GET endpoint for a route like:

`www.music.com/albums/[AlbumIdHere]/songs/[SongIdHere]`

```
const albumsRouter = express.Router();
const songsRouter = express.Router();

app.use('/albums', albumsRouter);

// ... routes for /albums

// This route represents /albums/:albumId/songs since we have already forwarded '/albums' above!
albumsRouter.use('/:albumId/songs', function(req, res, next) {
  req.albumId = req.params.albumId;
  next()
}, songsRouter);
```

```
// The route for handling a request to a specific track
songsRouter.get('/:songId', function(req, res, next) {
  let albumId = req.albumId; // Accesses the albumId we set earlier
  let songId = req.params.songId;

  // Get data here and send a response
  return res.send(`Album ${albumId} and track ${songId}`);
});
```

For more information, see **this blog post**.

Overview: How to Use Postman (20 minutes)

If you haven't yet, install **Postman** and watch as your instructor demonstrates its use with a locally served API.

Wrap-Up

If you did not finish the Babel/Node/Express tutorial, finish all parts as homework.

Begin work on the **Reddit.js** tutorial. Finish Parts 1-3 before class on Thursday.

Fill out our **Vibe Check form** with any feedback you have for the class.

Resources

How To Make Modular Routers in Express

Writing middleware for use in Express apps