Building & Testing an Express Application

Warm Up

- How are jQuery and JavaScript related?
- What do you know about node thus far? What have you used it for?
- In what order will the console log's appear in each example on the board?

What is Node?

Traditionally JavaScript is executed client-side, or in the browser on the consumers own computer. This is made possible by a browsers JavaScript Engine. Firefox's engine is called SpiderMonkey, and Chrome's is called V8.

Node, in it's most basic form, "is a JavaScript runtime built on Chrome's V8 JavaScript engine."

What is Node?

Node is a server-side platform developed by Ryan Dahl in 2009. Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript.

What is npm?

NPM (Node Package Manger) allows for organization of outside packages much like Ruby Gems. We'll cover it in more depth later on.

What is Express?

Express is a small framework built on top of the web server functionality provided by Node.js.

It helps to simplify and organize the server-side functionality (into an MVC architecture) of your application by providing abstractions over the more confusing parts of Node.js, and adding helpful utilities and features.

Why do we use Express?

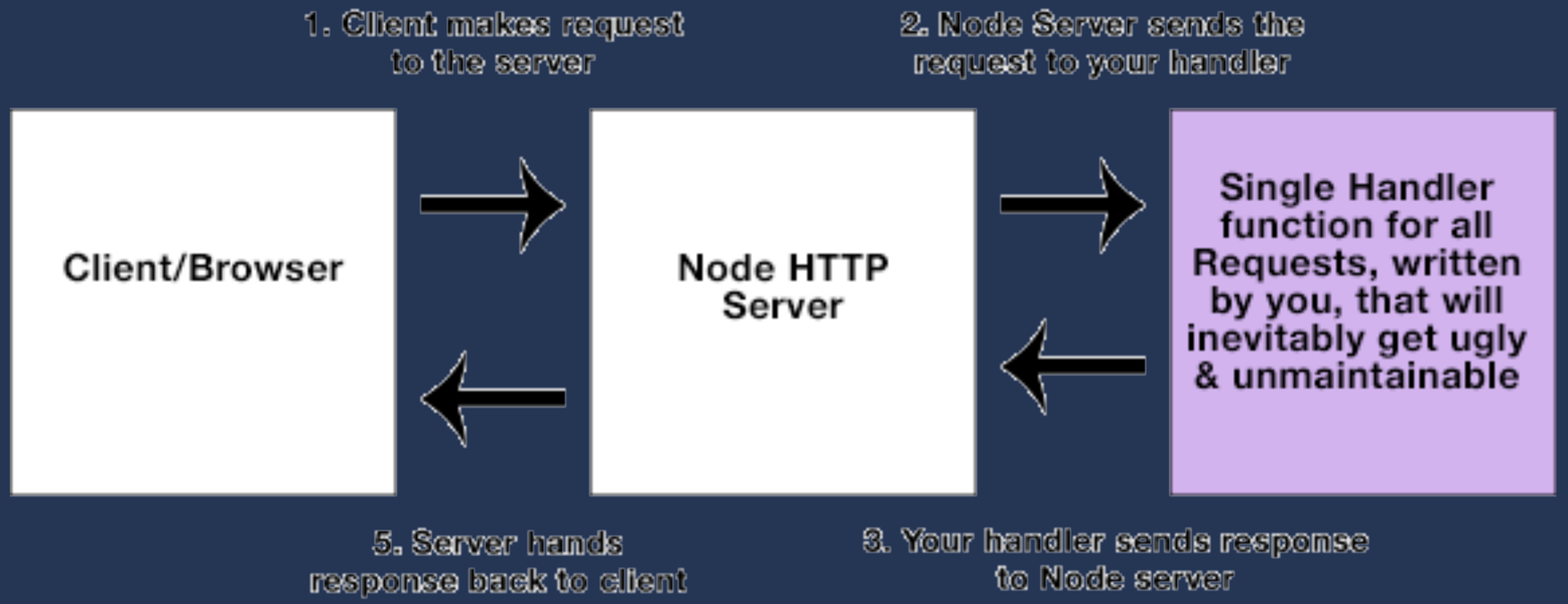
Think about how and why we use jQuery on the front-end. Vanilla JavaScript can be verbose and difficult to read. jQuery came along to give developers a nicer-looking syntax to perform the same operations.

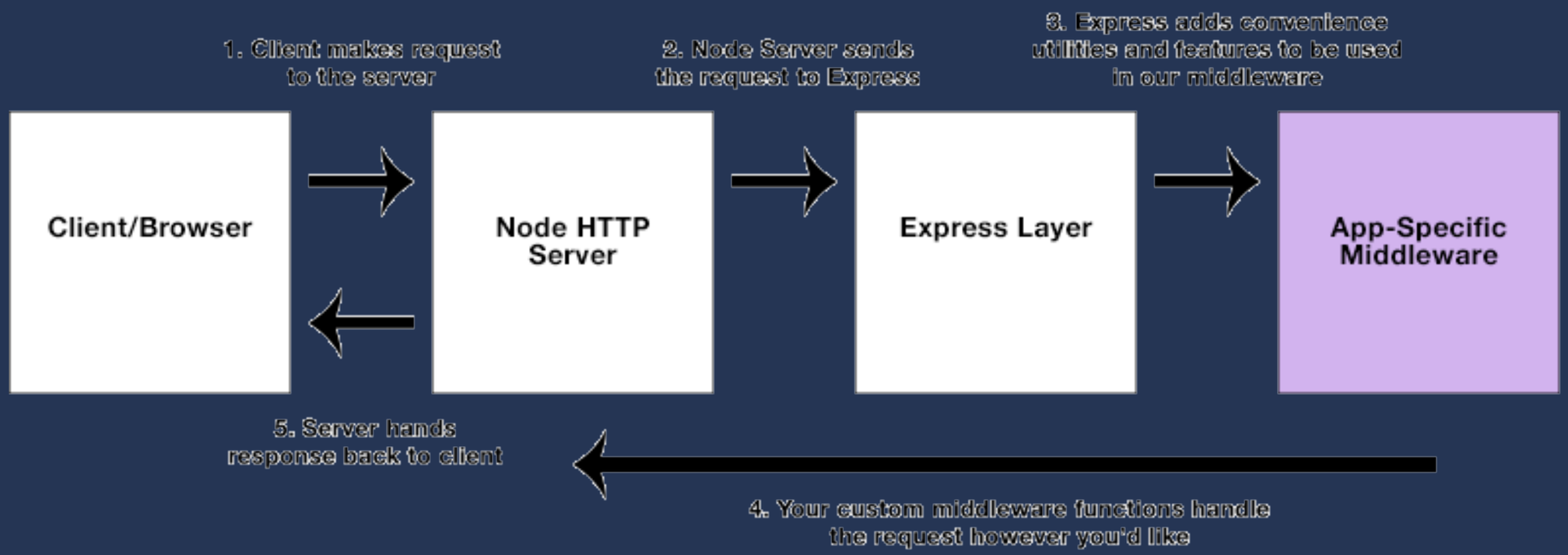
It was a library built to abstract the trickier parts of JavaScript and make them easier to write and work with. Express was built for very similar reasons.

Why do we use Express?

Just like browser-based JavaScript, the syntax for using plain Node.js isn't the friendliest.

Node gives you enough low-level features to build the back-end of an application, but Express is a light layer built on top of Node to make these low-level features a little easier to read and write.


1. Client makes request to the server
2. Node Server sends the request to your handler
Client/Browser
Node HTTP Server
Single Handler function for all Requests, written by you, that will inevitably get ugly & unmaintainable
5. Server hands response back to client
3. Your handler sends response to Node server


1. Client makes request to the server
2. Node Server sends the request to Express
3. Express adds convenience utilities and features to be used in our middleware
Client/Browser
Node HTTP Server
Express Layer
App-Specific Middleware
5. Server hands response back to client
4. Your custom middleware functions handle the request however you'd like

Routing & Middleware

Most of the Express code that you write will be routing middleware. Middleware is basically the "glue" between two systems that helps them work together (in our case, Node and Express).

Our code will be concerned with responding to client requests to different URLs with different methods (GET, POST, etc).

Routing & Middleware

```
app.get('/', function (request, response) {
  response.send('Hello World!')
})
```

This pattern is exactly how we can define and handle any routes in an Express application. There are four main pieces to this code:

- app - the instance of our Express application
- a METHOD - the method specified when the request is made from the client. (e.g. GET, POST, PUT, DELETE)
- a PATH - the endpoint that we are requesting
- a HANDLER - the function we write that contains the logic for how the request should be dealt with, and what kind of response it should return

Getting Started with Express

Let's build our app!!!