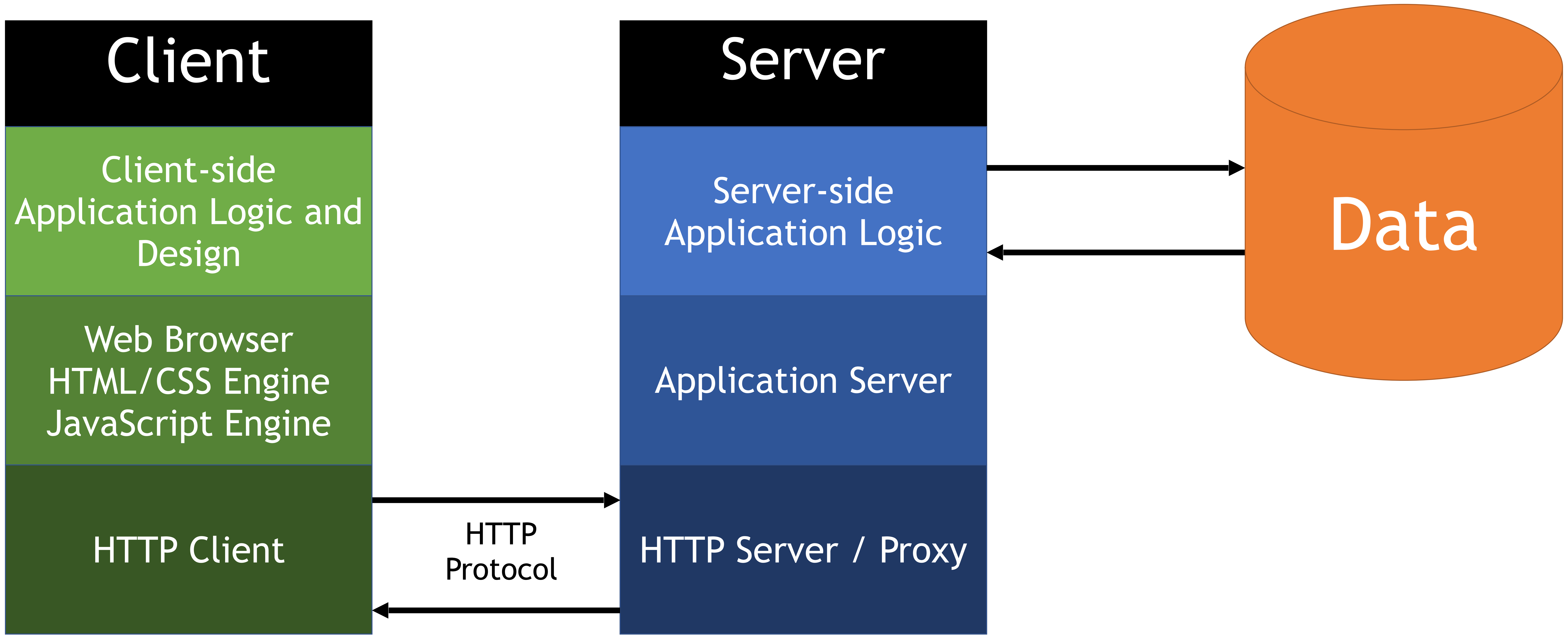
Review


Client
Client-side Application Logic and Design
Web Browser HTML/CSS Engine JavaScript Engine
HTTP Client
HTTP Protocol
Server
Server-side Application Logic
Application Server
HTTP Server / Proxy
Data

Review

- JavaScript Language Concepts (closures, prototypes, this binding)
- Client-side Web Application Technologies - HTML/CSS/JavaScript
- The HTTP Protocol - Requests, Responses, and their contents
- Server-side Web Application Concepts
    - Sources of request inputs (url path parameters, headers, body)
    - Routing (by method and request path), Middleware
    - Server-side response rendering and redirecting
- Database Integration

Review Doc

- Form groups of 2 or 3!
- One of you start a new Google doc that's a review study guide.
- We'll spend 10 minutes brain dumping in groups per subject and then time on Q&A and sharing notes

Topic #1 - HTML and the DOM

- What is the relationship between HTML and the DOM?
- How can your client-side script code access elements in the DOM? The more the better.
- Why is the significance of DOM events and how do you interact with them?

Topic #2 - HTML, the Browser, and HTTP

- What will be sent to the server when you click on an <a> link? Think as broadly (but also specifically) as possible.

- A single response from the server can lead to your browser making subsequent requests to the server. What are two useful, fundamentally different examples of this phenomena?

- How can an HTML document, without any JavaScript, enable a user to make a POST request from their browser? How does the developer control what information is sent to the server in the POST request?

Topic #3 - HTTP and The Server-side

- Why is routing based on both the HTTP method *and* URL useful? Think of at lease one example use case where it makes sense to have a route with the same URL and different methods.

- What are common sources of input from an HTTP request that a server app can use in its logic and response? Try to think of as many as possible with an example of how you accessed that input in an express route's req object.

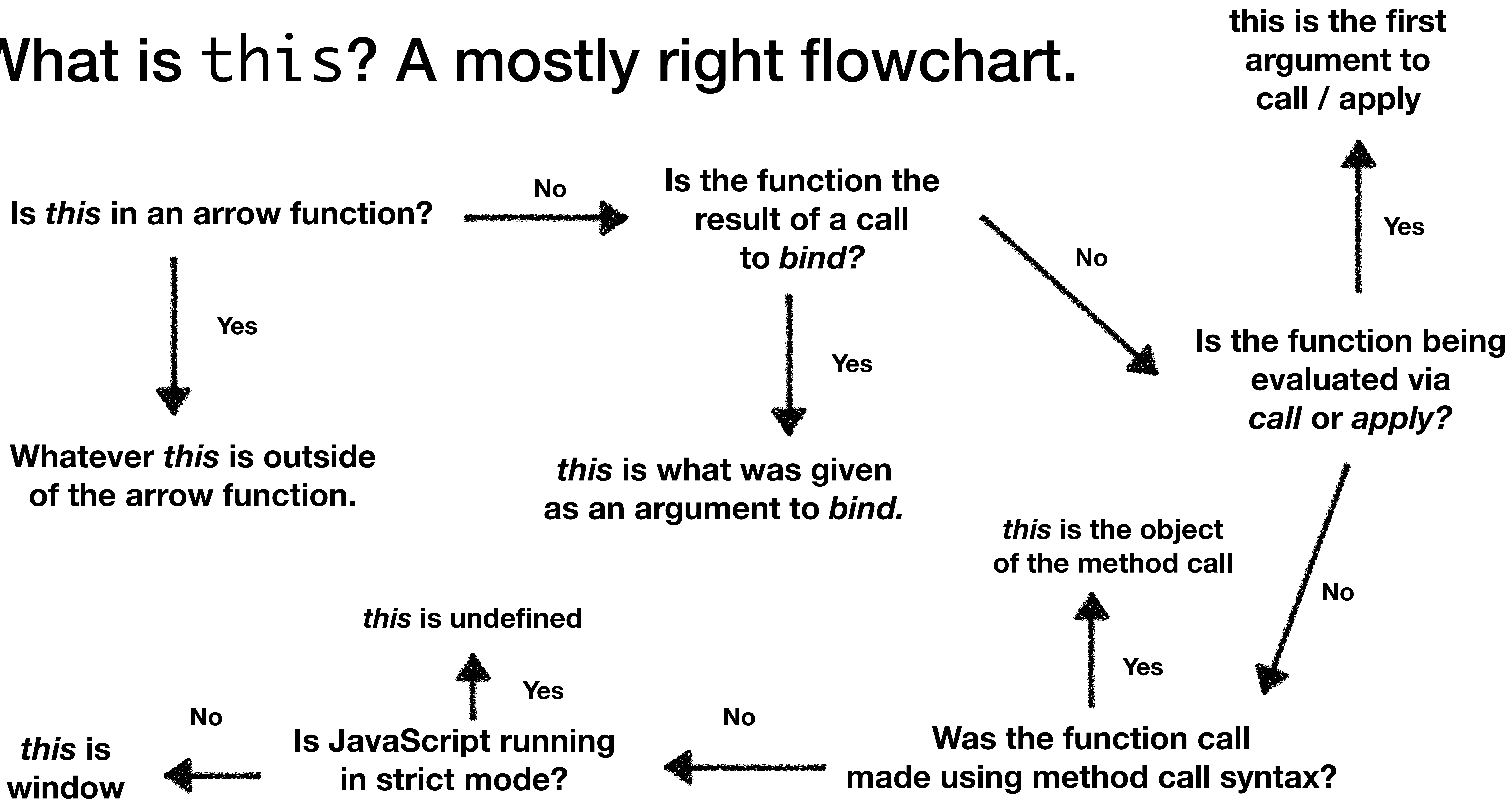
Topic #4 - Data Persistence

- Why is it bad practice to store application data in global variables? Why prefer another service, such as MySQL?
- In an application with user authentication, what should you store instead of the user's password in plaintext? How are you able to store something other than their password and still allow them to login?
- In an HTTP response, how can the server instruct the client to store data? How does the client present this data back to the server in a request? How is this capability useful?

What is this?

WHAT IS THIS?
THAT'S THE WHOLE JOKE: IT'S ABOUT JAVASCRIPT CONTEXT
nerator.net

What is `this`? A mostly right flowchart.


this is the first argument to call / apply
Is *this* in an arrow function?
No
Is the function the result of a call to *bind?*
No
Yes
Is the function being evaluated via *call* or *apply?*
Yes
Yes
Whatever *this* is outside of the arrow function.
this is what was given as an argument to *bind.*
this is the object of the method call
No
this is undefined
Yes
Yes
No
No
this is window
Is JavaScript running in strict mode?
Was the function call made using method call syntax?

Subjects to Continue Exploring

Advanced HTML / CSS

- Progressive Web Apps - Built sites that can be installed "like an app" on phones / tablets
- ARIA Accessibility - Improve access to people using assistive technologies
- Semantic HTML - Use tags and attributes that are convey meaning
- Responsive Images - Load images dependent on screen type
- Audio and Video tags for rich media embedded in pages
- CSS Grid Layout
- Z-index for content stacking control
- CSS Animations

HTML API Scripting

- fetch - make HTTP requests in JavaScript without page reloads
- localStorage - key-value persistent storage a web browser maintains
- indexeddb - document database in the web browser
- service workers - create rich offline client-side applications
- web workers - run computationally expensive JS tasks in background threads
- touch events - make client-side experiences better on phones/tablets
- web sockets - create a long lived, bi-directional connection to a server for "server push"
- Many more! https://developer.mozilla.org/en-US/docs/Web/API

Client-side Web Frameworks

- React (for views)
- Angular - Model and Views
- Vue.js - HTML, CSS, JS components

HTTP Protocol

- HTTPS (Secure Connection) Certificate
  - Explanation of HTTPS
  - Service to get a free certificate: https://letsencrypt.org/
  - Steps for using HTTPS in node.js/express
- Cross-Origin Resource Sharing - Share resources across domains
- Content Security Policy
- Caching - Overview of caching and related headers
- HTTP Authorization - Basic, Bearer
- Evolution of HTTP

Server-side Application Development

- Idempotence of GET, PUT, and DELETE HTTP method requests
- Implement RESTful APIs
- OAUTH2 for 3rd party identity management/integration - (Google/FB login)
- Multi-part request handling - allow file uploads in your web app
- Web sockets for rich communication between server and client
- Data validation and sanitization for user inputs

DevOps

- The 12 Factor Application
- Serverless Application Development (AWS Lambda, Netlify Functions)
- Containers and Docker
- Web Server / Reverse Proxy: nginx
- Web Caching Server: https://varnish-cache.org/