CS 4220

- Current Trends in Web Design & Development -

Prof. Cydney Auman

AGENDA

01 What is a Server?

02 Node.js Server

03 Express.js

04 Code Demo

What is a Server?

A **server** is a computer or a device that provides some functionality for other programs or devices, called "clients".

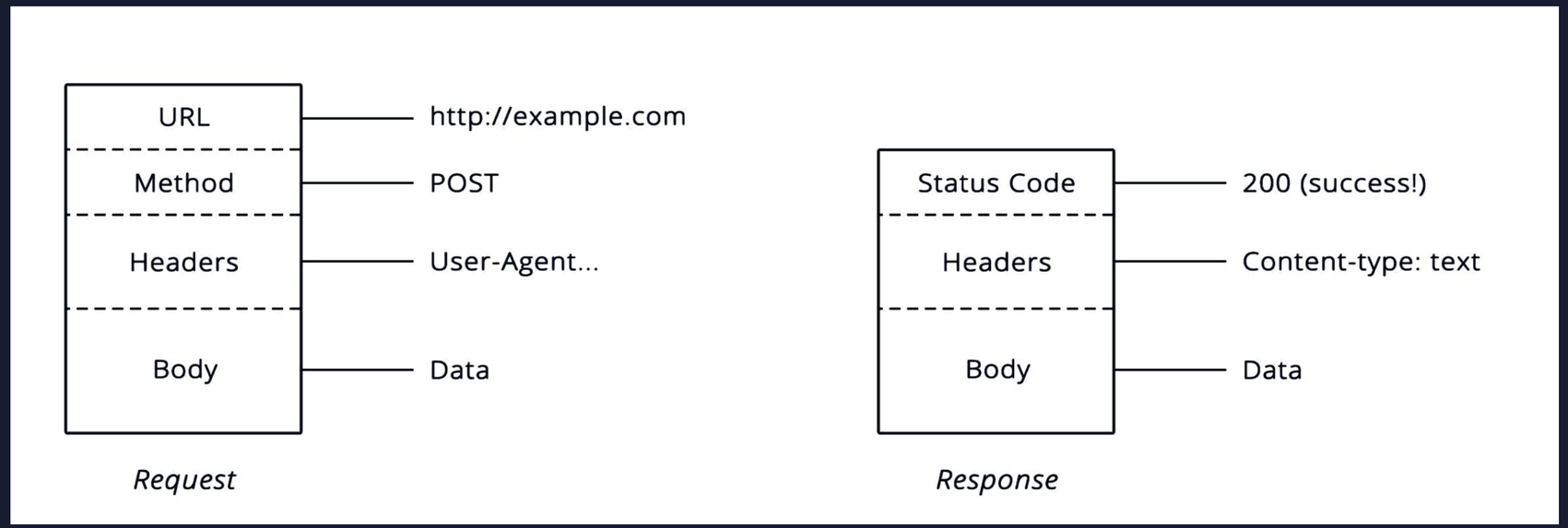
This architecture is called the client–server model. Servers provide various functionalities, often called "services", such as sharing data or resources. An API is an example of one such service that Servers can provide.

A single server can serve multiple clients, and a single client can use and access multiple servers.

Node.js Server

The built-in **http** module comes with a `createServer()` function and this allows us to easily spin up a simple server in Node.js.

`createServer((request, response))` accepts a function as an argument and this function is called every time a client connects to the server. The `request` and `response` are objects representing the incoming and outgoing data.



Request and Response

- **request** is the first argument and comes in from the client such as a browser or CLI - this sometimes shortened to req.

The request argument contains information about the request, such as its url, http method, headers, body and etc.

- **response** is the second argument in the function. Just like the prior argument is often shortened to res.

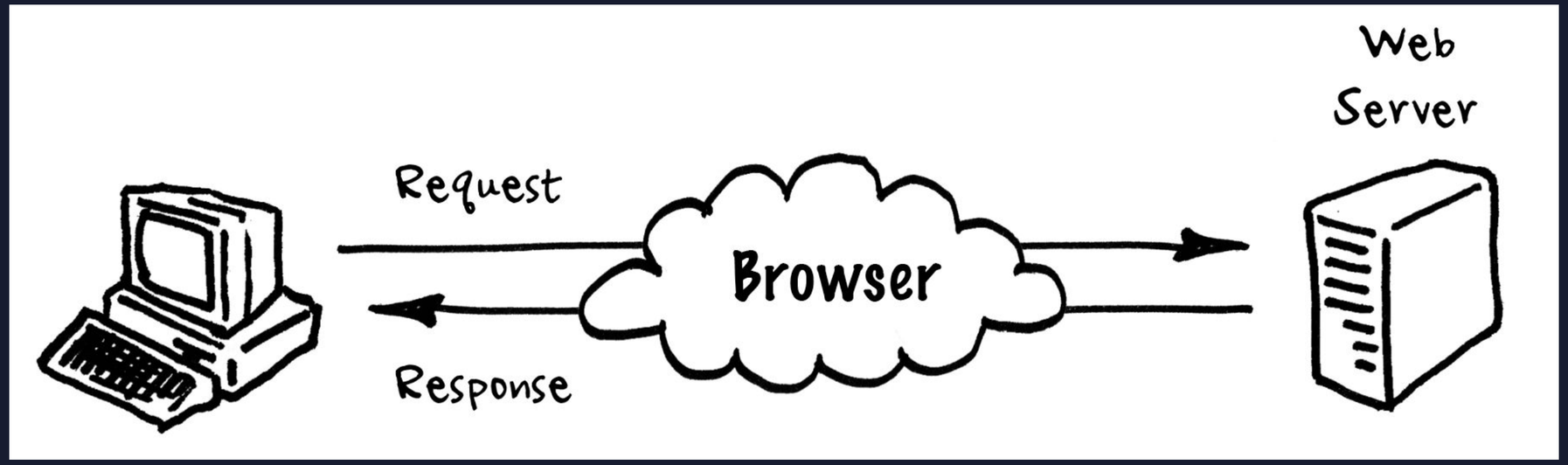
Each time a request comes in - some data is prepared and then data needs to be sent. In Node.js we can do the by calling functions on the response object. (`response.send()` or `response.json()`).

Eventually, you must call one of these methods. As these methods do the actual sending of data. If one of these methods is not called, the server request will just hang forever.

Node.js Server

By building an http server in Node.js, we will be able to access it locally via the browser. When we open our server page in the browser, this will then send a request to our own computer which is running the Node.js server.

Making this request will causes the server function to run. The server will process incoming information from the request and then send back a response. And we will be able to view this response in the browser.



What is Express.js?

Express.js describes itself as a "Fast, unopinionated, minimalist web framework for Node.js”
It also provides a robust set of features for building single and multi-page as well as hybrid web applications.

Express.js uses built-in utilities from Node.js under the hood. But it is able to provides a set of common handlers, settings, utilities, and other tools which greatly improve the experience and speed of creating a web server.

The Express.js framework comes with the basics of what is needed to get started. And it also doesn’t impose any strict rules on how you use those tools to create your Node.js application.

Why use Express.js?

As we've seen Node.js provides a solid environment and foundation for building various applications, but in order to fully realize the potential we often rely on a community-supported ecosystem. This is exactly what Express.js offers - a strong open-source community which provides countless extensions in the form of "middleware".

Middleware are libraries that help our Servers work with cookies, sessions, user logins, URL parameters, POST data, security headers, and much more.

Having common tools combined with the rich ecosystem is what has made Express.js the most popular server framework for Node.js.

Using nodemon with Servers

nodemon is an npm module and tool that helps with development of Node.js based applications by automatically restarting the Node application when file changes in the directory are detected.

nodemon does not require any additional changes to our code or method of development. In fact nodemon can be easily installed using npm and then adding a line our scripts object inside the package.json file.

```
console.log('Week 11');
console.log('Code Examples');
```

Review and Prep

Review

- Review Slides
- Review and Run Code Examples

Preparation for Next Class

- Read Chapter 20 - Node.js (HTTP Servers)
- Read about Express Middleware
  https://expressjs.com/en/guide/using-middleware.html