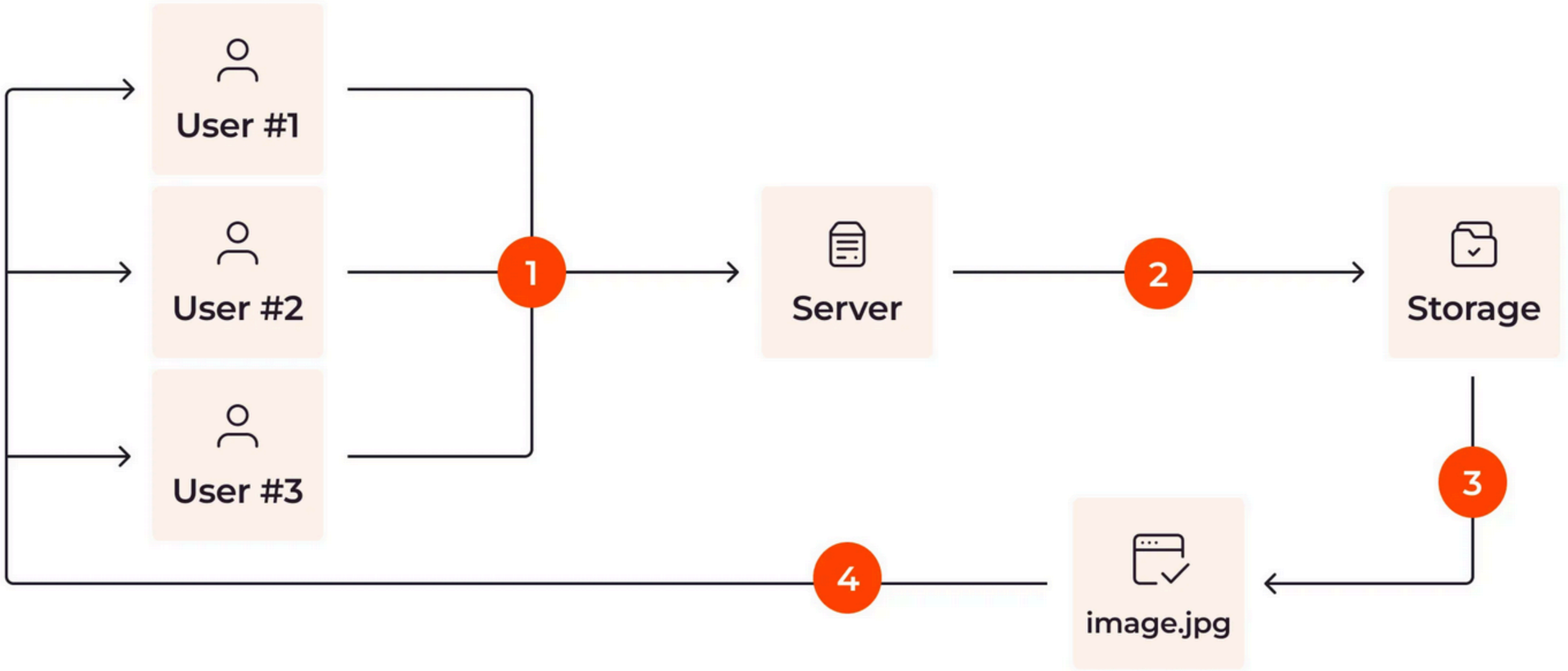
Serving Dynamic Content

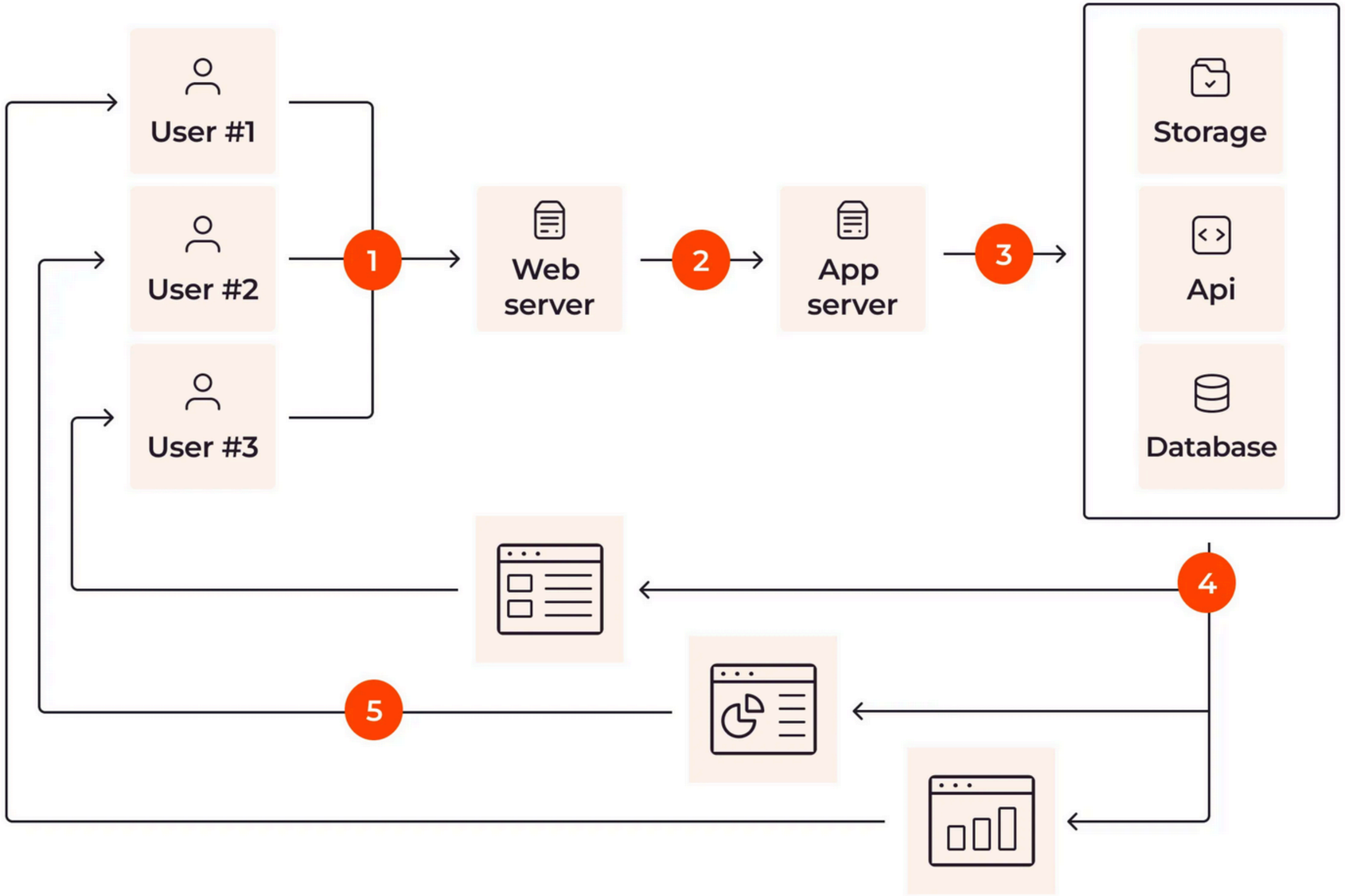
Node.js

Static Content

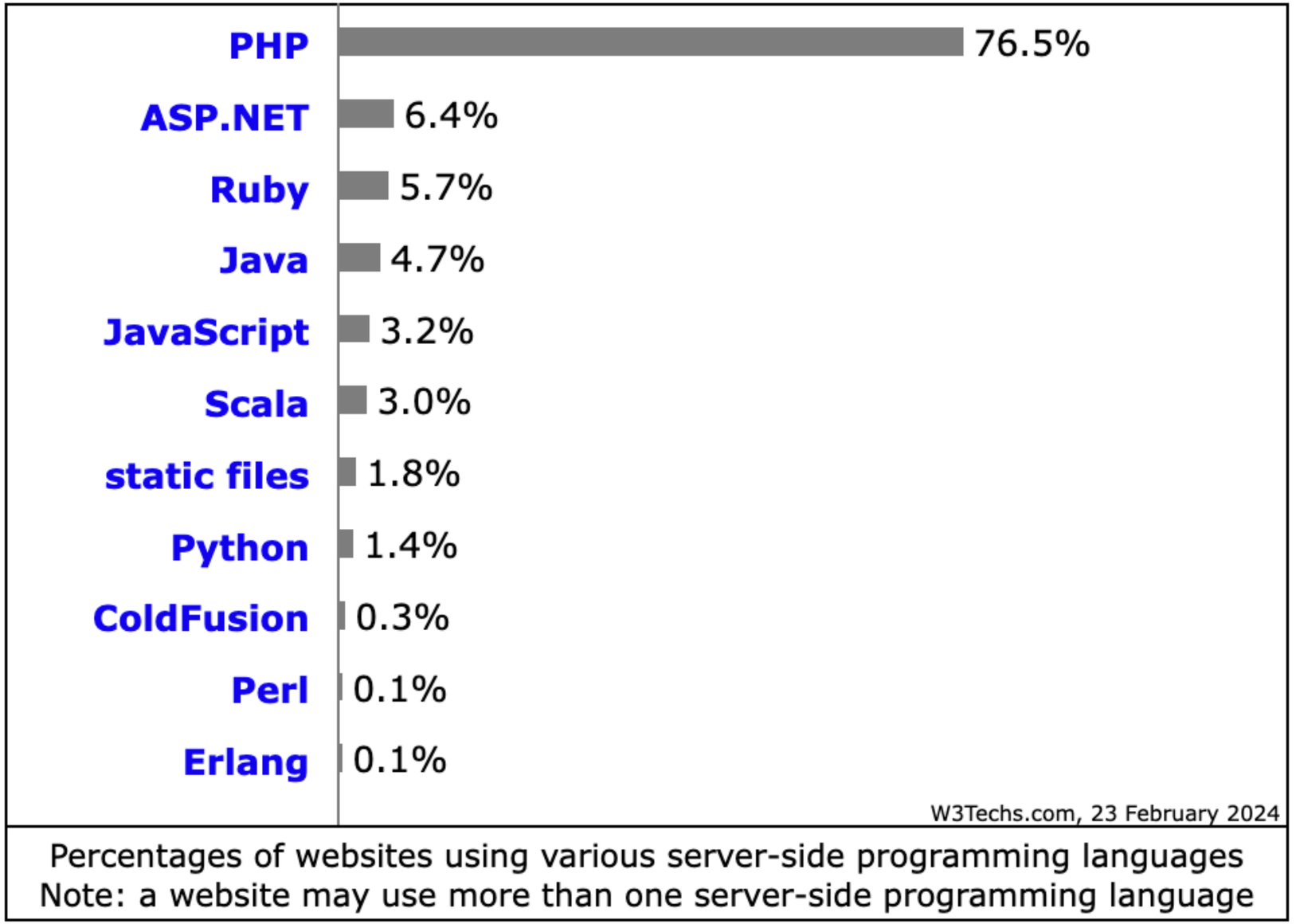


https://gcore.com/learning/static-vs-dynamic-content-understanding-the-difference/

Dynamic Content



Top Languages for "server side scripting"



1995: PHP is dead, learn ColdFusion
2002: PHP is dead, learn ASP.net
2003: PHP is dead, learn Django
2004: PHP is dead, learn Ruby on Rails
2010: PHP is dead, learn Flask
2011: PHP is dead, learn AngularJS
2016: PHP is dead, learn Next.js
2022: PHP is dead, learn Python
2023:

PHP

Not

Jack Forge @TheJackF... · 13h

In 2023.

28 people learned how to write code in **PHP**.

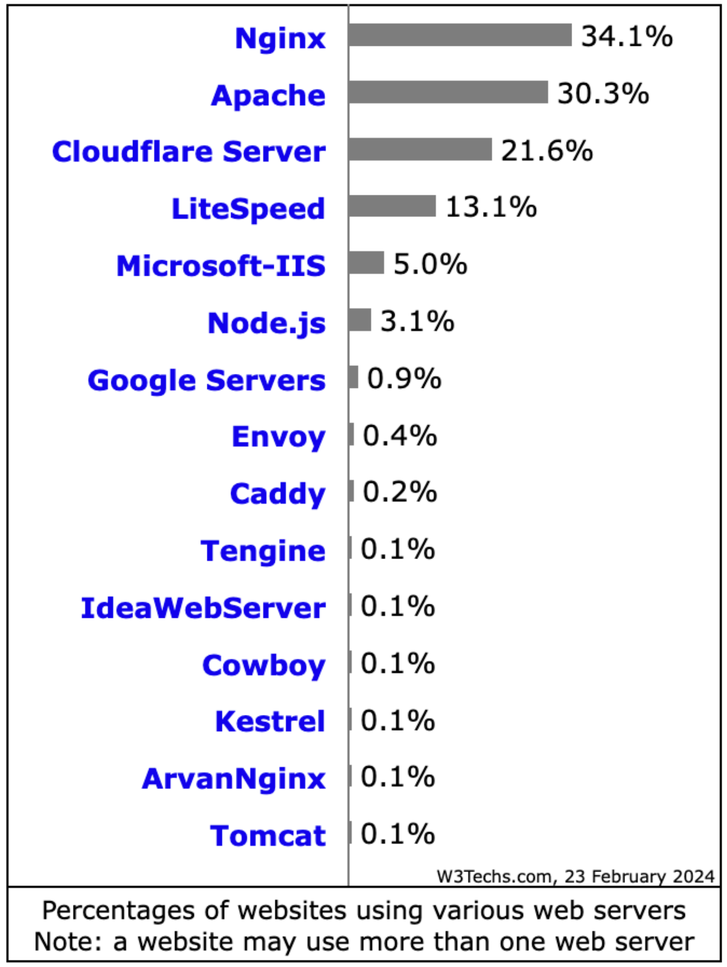
Every single of them is now a millionaire.

Let that sink in.

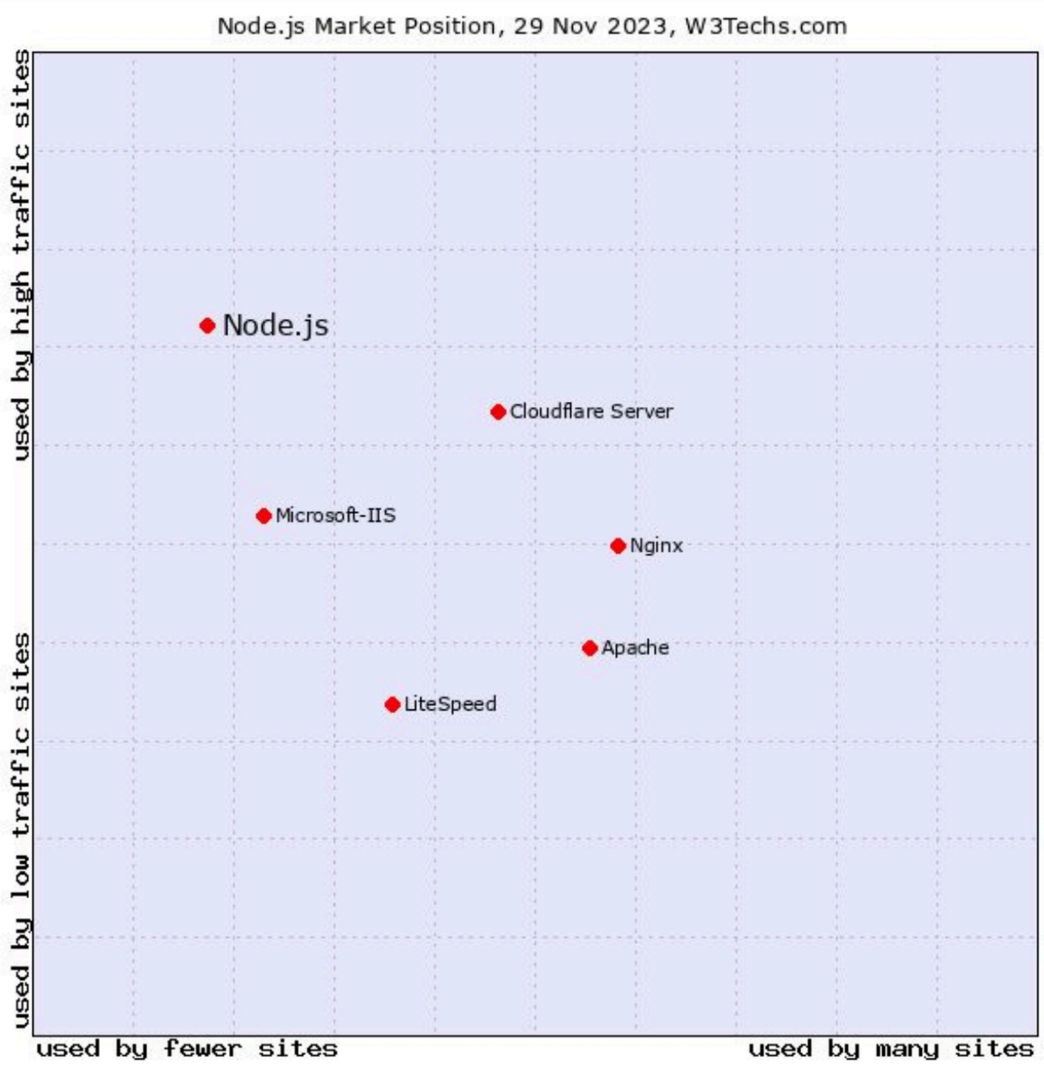
69 39 654 91K

Web Servers

Node.js



Percentages of websites using various web servers
Note: a website may use more than one web server



- "Node.js is an open-source, cross-platform runtime environment for JavaScript that executes the code outside of the browser." https://radixweb.com/blog/nodejs-usage-statistics
- "Node.js is designed to build scalable network applications." https://nodejs.org/en/about
- Node.js is system we will use in this class to build and deploy dynamic web sites Geoff

Lots of stuff is built into Node

Similar to all of the packages shipped with Java

https://www.w3schools.com/nodejs/ref_modules.asp

Module	Description
assert	Provides a set of assertion tests
buffer	To handle binary data
child_process	To run a child process
cluster	To split a single Node process into multiple processes
crypto	To handle OpenSSL cryptographic functions
dgram	Provides implementation of UDP datagram sockets
dns	To do DNS lookups and name resolution functions
domain	Deprecated. To handle unhandled errors
events	To handle events
fs	To handle the file system
http	To make Node.js act as an HTTP server
https	To make Node.js act as an HTTPS server.
net	To create servers and clients
os	Provides information about the operation system
path	To handle file paths
punycode	Deprecated. A character encoding scheme
querystring	To handle URL query strings
readline	To handle readable streams one line at the time
stream	To handle streaming data
string_decoder	To decode buffer objects into strings
timers	To execute a function after a given number of milliseconds
tls	To implement TLS and SSL protocols
tty	Provides classes used by a text terminal
url	To parse URL strings
util	To access utility functions
v8	To access information about V8 (the JavaScript engine)
vm	To compile JavaScript code in a virtual machine
zlib	To compress or decompress files

everything else: Node Package Manager

npm

- You can do almost everything in node without any extra "packages" but why?
  - Java with only the classes in java.lang
    - Sometimes you need more (e.g. connect to PostgreSQL, or Mongo)
- Get extra packages using "node package manager"
  - UNIX> npm install XXXX
    - XXXX is the name of the package you want to install
    - other common: update, uninstall, version (versions of default packages) , list (shows installed packages and versions)

- Almost always "npm install express"
  - other packages as needed

https://expressjs.com/en/5x/api.html

- Must install packages in each directory in which you run a node instance
    - WHY?

Node: Hello World

- create directory and CD into it
- Determine the "port" you are going to use
  - http://165.106.10.133:30006/index6.html
- npm install express
- make the file
  - hw.js
- start Node with the file
  - UNIX> node hw.js
- In a browser navigate to
  - http://165.106.10.133:30001/helloworld

```
const express = require('express')
const app = express()
const port = 30001

app.use("/helloworld", function (req, res) {
    res.write("<html><body>")
    res.write("Hello World")
    res.end("</body></html>")
})

app.listen(port, function (error) {
    if (error) throw error
    console.log(`Server created on port ${port}`)
})
```

prints to terminal

/home/gtowell/Private/383/NodeIntro24/hw.js

Making it active

and serving actually static pages

- create directory and CD into it
  - if needed
- Determine the "port"
- npm install express
  - if needed
- make the file
  - hw2.js
- start Node with the file
  - node hw2.js
- In a browser navigate to
  - http://165.106.10.133:30001/helloworld
  - http://165.106.10.133:30001/hello.txt

/home/gtowell/Private/383/NodeIntro24/hw2.js

```
const express = require('express');
const app = express();
let port = 30001;

app.use("/", express.static(__dirname))

let counter=0

app.use("/helloworld", function (req, res) {
    console.log(req)
    res.write("<html><body>");
    res.write("Hello World");
    res.write(`Hit Counter: ${counter++}`)
    res.end("</body></html>");
});

app.listen(port, function (error) {
    if (error) throw error
    console.log(`Server created on port ${port}`)
});
```

Reading Parameters

ONLY parses JSON!
As from a JS method on client page

ONLY parses k-v pairs
As from an html form

- GET and POST handled differently
- GET is easy to test from browser
  - just edit URL
- POST -- Either need form / javascript in browser to submit post or a tool like Postman

/home/gtowell/Private/383/NodeIntro24/post_get.js

```
const express = require('express');
const app = express();
let url = require('url');
let port = 30001

app.use(express.json());
app.use(express.urlencoded());

app.use("/post-test", function (req, res) {
    writePage(res, req.body)
})

app.use("/get-test", function (req, res) {
    let url_parts = url.parse(req.url, true);
    let query = url_parts.query;
    writePage(res, query)
})


function writePage(res, params) {
    res.write('<html><body>')
    writeParams(res, params);
    res.end("<br>Response Post</body></html>");
}

function writeParams(res, params) {
    for (const [k, v] of Object.entries(params)) {
        res.write(`<br>${k}:${v}\n`);
    }
}

app.listen(port, function(error){
    if(error) throw error
    console.log(`Server created on port ${port}`)
})
```

Short and Long Polling

- Short -- respond immediately*
- Long -- respond when you have something to say

```
const express = require('express');
const app = express();
const url = require('url');
const port = 30001

let counter = 0;
app.use(express.json());
app.use(express.urlencoded());

app.use("/post-test", function (req, res) {
    let tt = Math.random() * 5000;
    console.log(tt)
    setTimeout(() => { writePage(res, req.body) }, tt)
})

// stuff not shown

const server = app.listen(port, function(error){
    if(error) throw error
    console.log("Server created Successfully")
})

server.setTimeout(2500, (socket) => {
    console.log(socket);
    socket.destroy();
  });
```

Simulate waiting for new information

Kill connection if too long (2.5 sec)
Long poll null return

/home/gtowell/Private/383/NodeIntro24/post_get2.js