NODE JS

Md Abdul Ahad Linkon - Software Engineer

12/ 2025

JOIN US

TABLE OF CONTENT

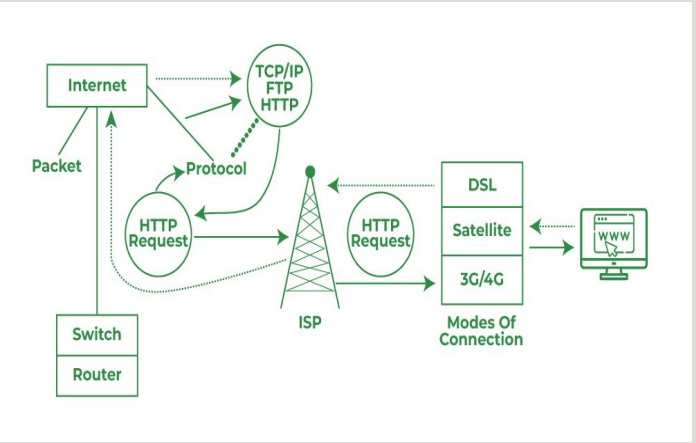
1. **Internet**
2. **Frontend Basics**
3. **LANGUAGE**
4. **Version Control Systems**
5. **Repo Hosting Services**
6. **Relational Databases**
7. **APIs**
8. **Caching**
9. **Web Servers**
10. **Databases**
11. **Testing**
12. **Docker**
13. More - (**Architectural Patterns, Design and Architecture, System design, Real time data**)
    a. Scaling database
    b. non-sql database

How does the Internet Work?

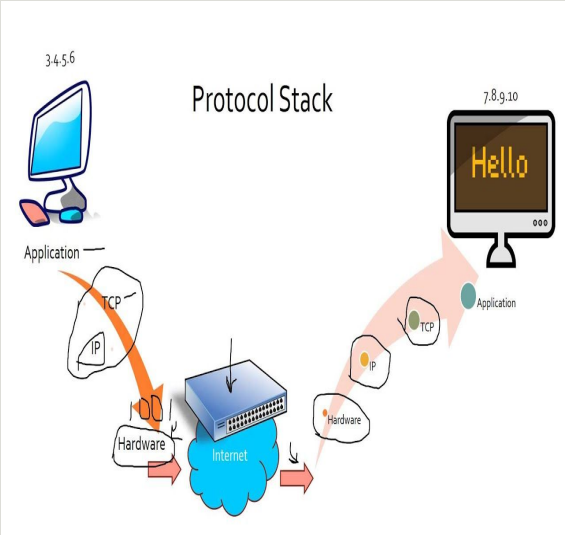
- The internet is a global network of interconnected computers that uses a standard set of communication protocols to exchange data.

---

https://cs.fyi/guide/how-does-internet-work |
https://www.youtube.com/watch?v=x3c1ih2NJ

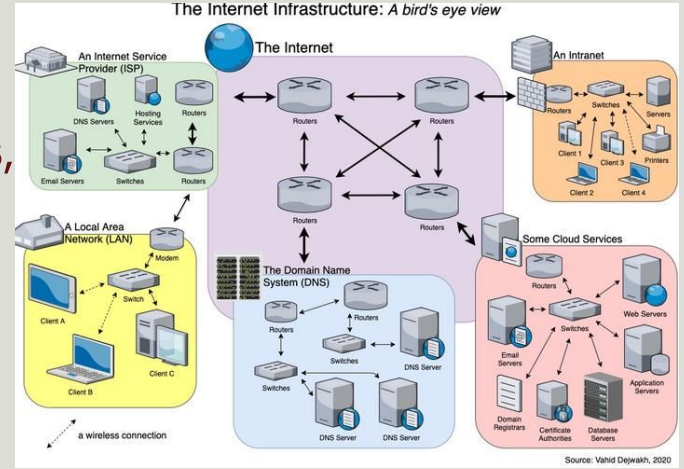


The basic concept of Intern:



Answer. To understand the internet, it's important to be familiar with some basic concepts and terminology. Here are some key terms and concepts to be aware of:

Packet , router,
ip address,
domain name, dns,
http, https, ssl, tsl

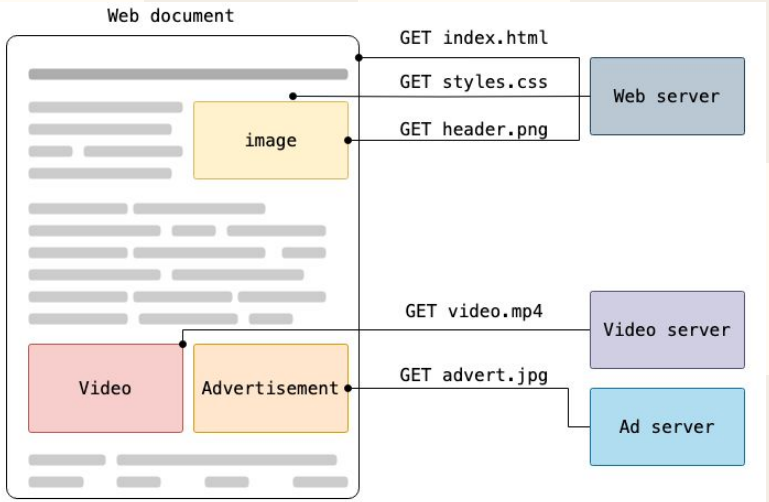


What is HTTP

Internet

The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web, and is used to load webpages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack. A typical flow over HTTP involves a client machine making a request to a server, which then sends a response message.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview



▼ Request Headers

:authority: www.google.com
:method: GET
:path: /
:scheme: https
accept: text/html
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0

▼ Response Headers

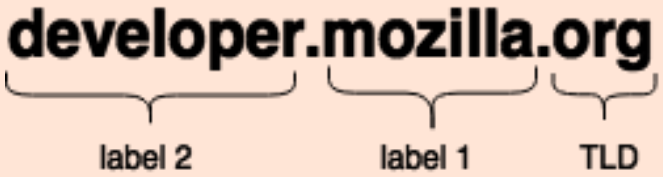
cache-control: private, max-age=0
content-encoding: br
content-type: text/html; charset=UTF-8
date: Thu, 21 Dec 2017 18:25:08 GMT
status: 200
strict-transport-security: max-age=86400
x-frame-options: SAMEORIGIN

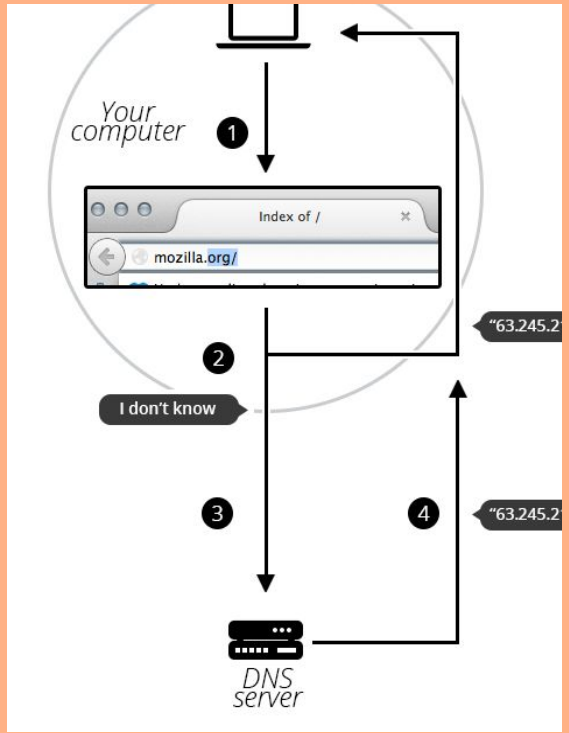
What is a Domain Name?

Structure of domain names

TLDs tell users the general purpose of the service behind the domain name. The most generic TLDs (`.com`, `.org`, `.net`) don't require web services to meet any particular criteria, but some TLDs enforce stricter policies so it is clearer what their purpose is. For example:

- Local TLDs such as `.us`, `.fr`, or `.se` can require the service to be provided in a given language or hosted in a certain country — they are supposed to indicate a resource in a particular language or country.
- TLDs containing `.gov` are only allowed to be used by government departments.
- The `.edu` TLD is only for use by educational and academic institutions.

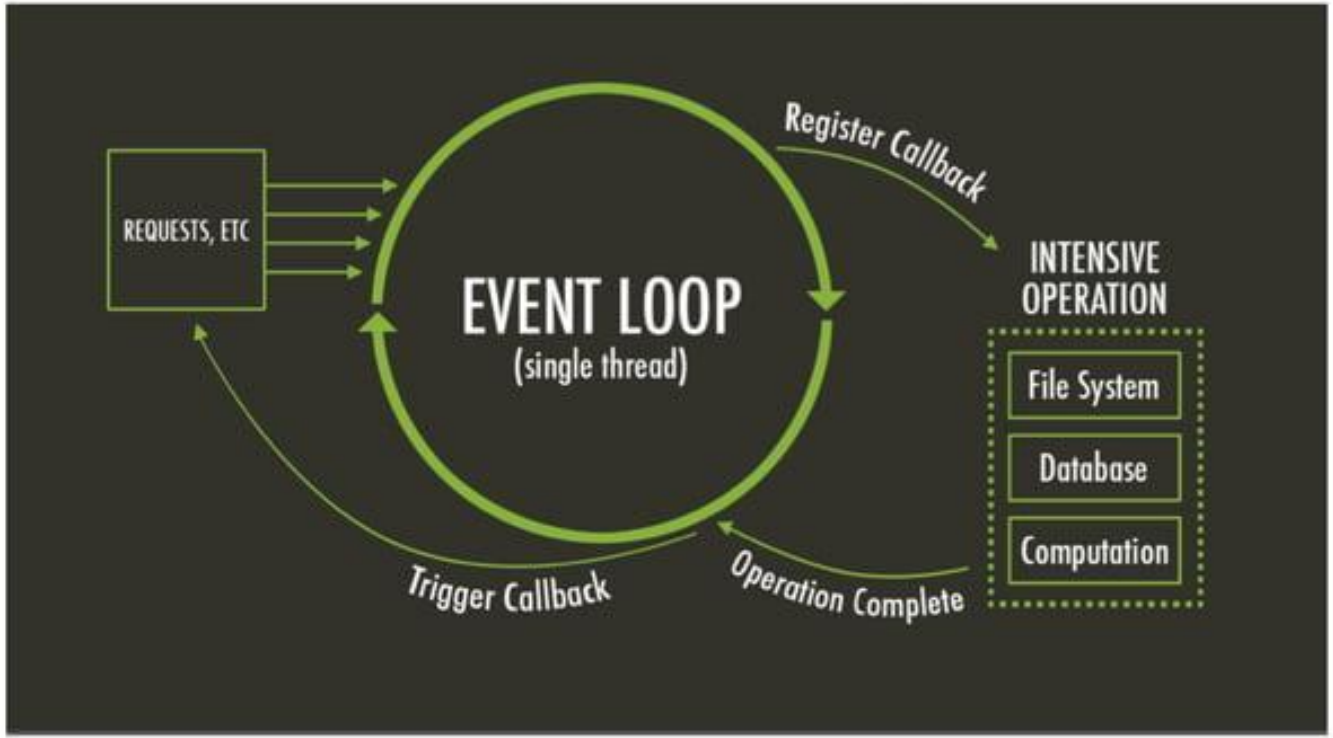


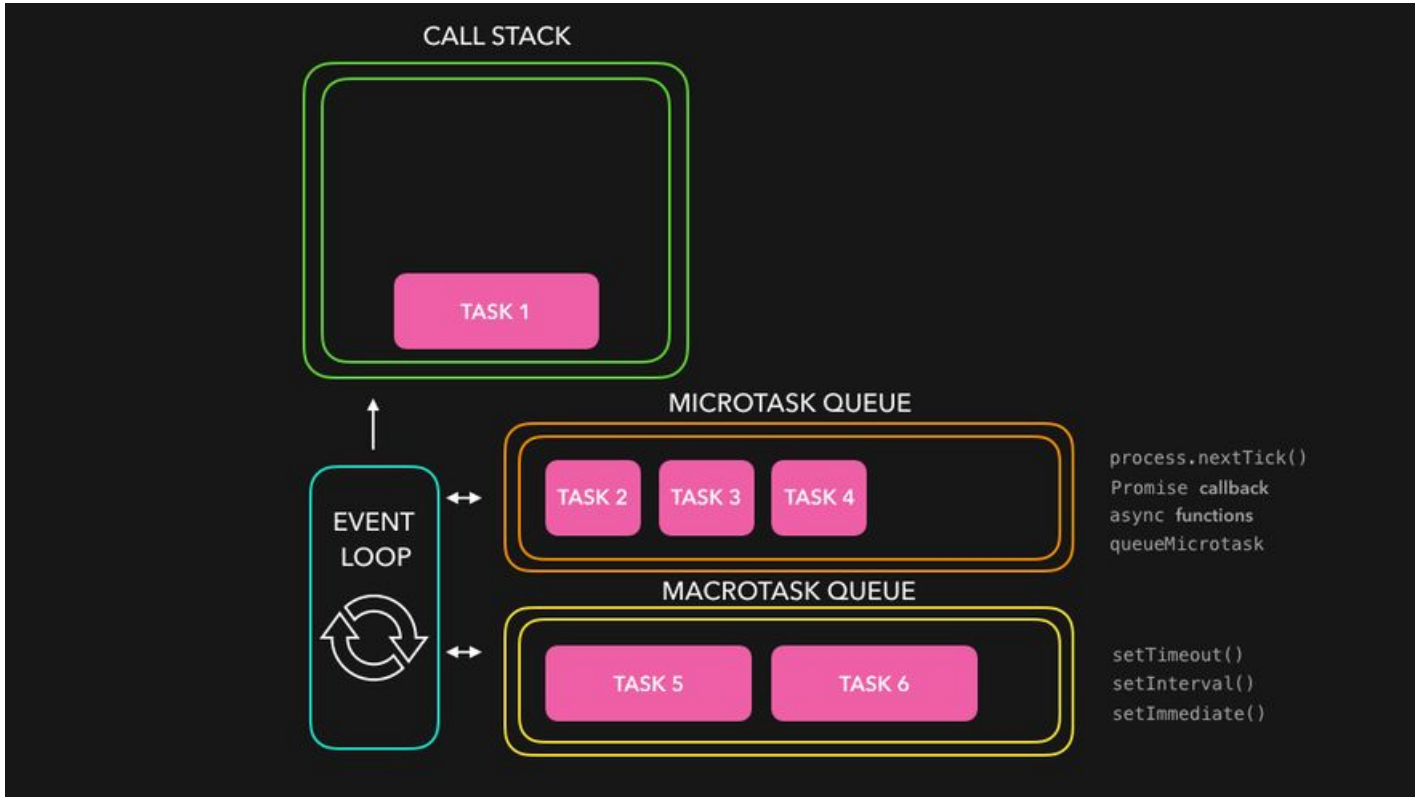


https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Web_mechanics/What_is_a_domain_name#structure_of_domain_names

Node.js Foundation

Node.js Event loop




CALL STACK
TASK 1
MICROTASK QUEUE
TASK 2
TASK 3
TASK 4
process.nextTick()
Promise callback
async functions
queueMicrotask
EVENT LOOP
MACROTASK QUEUE
TASK 5
TASK 6
setTimeout()
setInterval()
setImmediate()

Blocking I/O

```
1  var users = db.query("SELECT * FROM users")
2  console.log('All users:', users);
3
4  var posts = db.query("SELECT * FROM post")
5  console.log('All posts:', posts);
6
```

Blocking I/O

```
1  var users = db.query("SELECT * FROM users")
2  console.log('All users:', users);
3
4  var posts = db.query("SELECT * FROM post")
5  console.log('All posts:', posts);
6
```

Blocking I/O

```
1  var posts = db.query("SELECT * FROM post")
2  console.log('All posts:', posts);
3
4  var users = db.query("SELECT * FROM users")
5  console.log('All users:', users);
6
```

Sum = 30 + 10

Non-Blocking I/O

```
1 var posts = db.query("SELECT * FROM post", function(posts) {
2   console.log('All posts:', posts);
3 })
4
5 var users = db.query("SELECT * FROM users", function(users) {
6   console.log('All users:', users);
7 })
```

Non-Blocking I/O

```
1  var posts = db.query("SELECT * FROM post", function(posts) {
2    console.log('All posts:', posts);
3  })
4
5  var users = db.query("SELECT * FROM users", function(users)
6    console.log('All users:', users);
7  })
```

min=10
max=30

Asynchronous programming techniques

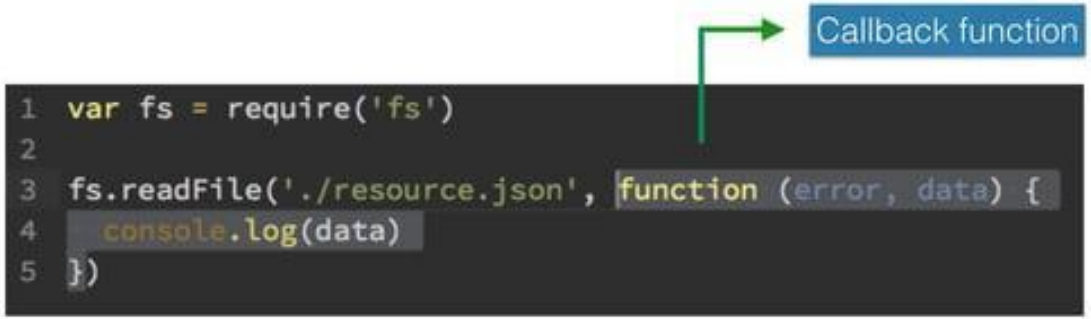
The two model of Async Processing

- **Callback function** generally define logic for one-off responses. – Ex: perform a database query

- **Event listeners,** are essentially callbacks that are associated with a conceptual entity (an event). – Respond to repeating events – EX: HTTP server emits a request event when an HTTP request is made

What is callback function?

- A **callback** is a function, passed as an argument to an **asynchronous** function
- It describes what to do **after** the **asynchronous** operation has completed



What is event-listeners?

```
1  //jQuery
2  $("#dataTable tbody tr").on("click", function() {
3    console.log( $( this ).text() );
4  });
5
6  $("div").on("mouseenter mouseleave", function() {
7      console.log("mouse hovered over or left a div");
8  });
9
```

What is event-listeners?

```
10 //Node.js
11
12 var req = http.request({hostname:'www.google.com'}, function(res) {
13     res.setEncoding('utf8');
14 });
15
16 req.on('response', function (response) {
17
18     var data = "";
19
20     response.on('data', function (chunk) {
21         console.log(chunk);
22         data += chunk;
23     });
24
25     response.on('end', function(){
26         callback(data);
27     })
28
29 });
```

Stream

```
1  var fs = require('fs');
2  var readableStream = fs.createReadStream('japan.avi');
3  var data = '';
4
5  readableStream.on('data', function(chunk) {
6      data+=chunk;
7      //compress file
8      //convert to other extension
9      //upload to backup server
10 });
11
12 readableStream.on('end', function() {
13     console.log(data);
14 });
```

When to use Node.js?

- Creating streaming based real-time services, web chat applications, static file servers etc.
- Applications that have a lot of **concurrent** connections and each request only needs **very few CPU cycles**

The Essence of Node

- JavaScript on the Server
- More than server-side JavaScript
- Fabulous framework
- Asynchronous programming
- Module-driven development

It's is JavaScript

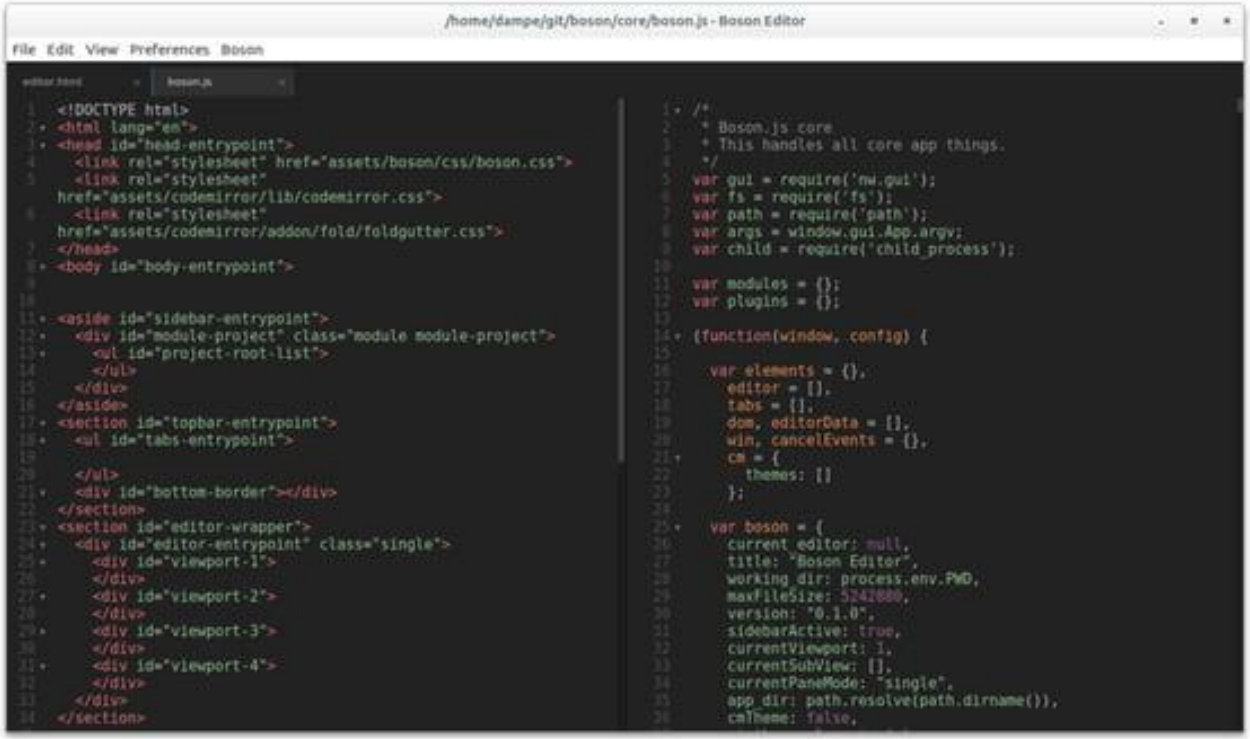
- Easy to learn
- Lets you unify your client/server logic
- Productivity
- Already in the Enterprise (Paypal, Netflix, Walmart, IBM)

More than server-side JavaScript

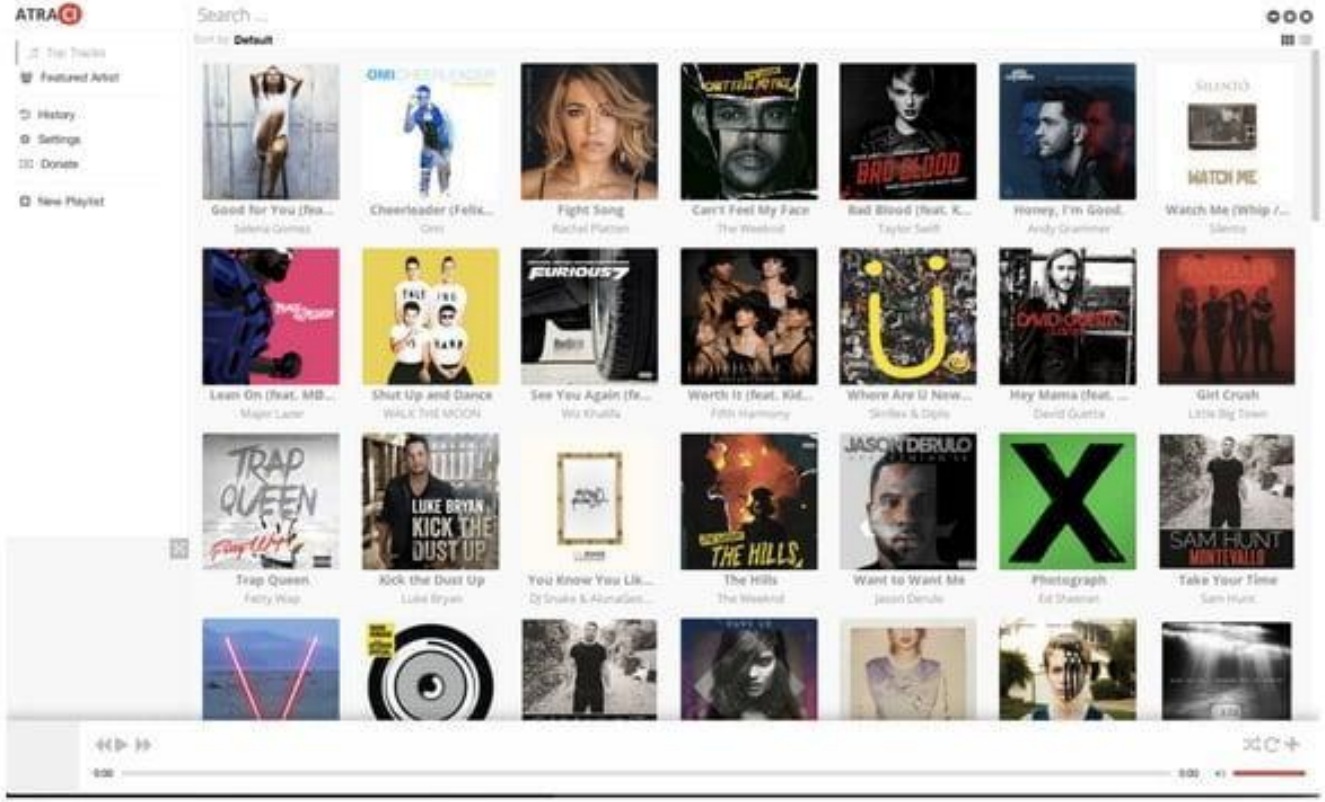
- Web server
- Robot controller (tessel.io)
- Command line application
- Proxy server
- Music machine
- Desktop application tooling: NW.js

Desktop Application (torrent)

Desktop Application (editor)



Desktop Application (music)



Desktop Application (game)

Fabulous framework

- Express
- Restify (for building REST APIs, automatic DTrace support)
- Hapi (configuration-centric framework)
- Sails (fast production-ready)
- Meteor (realtime application)

Module-driven development

- Node Package Manager (NPM)
- 180,000++ packages
- 90,000,000++ downloads per day
- Simplicity
- Decoupled and reusable coding

```
module.exports = function archy (obj, prefix, opts) {
    if (prefix === undefined) prefix = '';
    if (!opts) opts = {};
    var chr = function (s) {
        var chars = {
            '│' : '│',
            '└' : '`',
            '├' : '+',
            '─' : '-',
            '┬' : '-'
        };
        return opts.unicode === false ? chars[s] : s;
    };

    if (typeof obj === 'string') obj = { label : obj };

    var nodes = obj.nodes || [];
    var lines = (obj.label || '').split('\n');
    var splitter = '\n' + prefix + (nodes.length ? chr('│') : ' ') + ' ';

    return prefix
        + lines.join(splitter) + '\n'
        + nodes.map(function (node, ix) {
            var last = ix === nodes.length - 1;
            var more = node.nodes && node.nodes.length;
            var prefix_ = prefix + (last ? ' ' : chr('│')) + ' ';

            return prefix
                + (last ? chr('└') : chr('├')) + chr('─')
                + (more ? chr('┬') : chr('─')) + ' '
                + archy(node, prefix_, opts).slice(prefix.length + 2)
            ;
        }).join('')
    ;
};
```

```
├── async@0.9.0
├── basic-auth-connect@1.0.0
├─┬ body-parser@1.0.0
│ ├── qs@0.6.6
│ └─┬ raw-body@1.1.7
│   ├── bytes@1.0.0
│   └── string_decoder@0.10.31
├─┬ bundle-up3@1.1.1
│ ├── async@0.1.22
│ ├─┬ autoprefixer-stylus@0.5.0
│ │ ├─┬ autoprefixer-core@4.0.2
│ │ │ ├── caniuse-db@1.0.30000092
│ │ │ └─┬ postcss@3.0.7
│ │ │   ├── js-base64@2.1.7
│ │ │   └─┬ source-map@0.1.43
│ │ │     └── amdefine@0.1.0
│ │ └─┬ multi-stage-sourcemap@0.2.1
│ │   └─┬ source-map@0.1.43
│ │     └── amdefine@0.1.0
│ ├─┬ clean-css@0.8.3
│ │ └─┬ optimist@0.3.7
│ │   └── wordwrap@0.0.2
│ ├── lodash@0.9.2
│ ├── mkdirp@0.3.5
│ ├─┬ stylus@0.19.8
│ │ ├── cssom@0.2.0
│ │ ├── growl@1.1.0
│ │ └── mkdirp@0.0.7
│ └── uglify-js@1.2.6
├─┬ bunnycron@0.2.7 -> /Users/Saranyhat/Jitta/git/bunnycron
```

```
module.exports = function archy (obj, prefix, opts) {
    if (prefix === undefined) prefix = '';
    if (!opts) opts = {};
    var chr = function (s) {
        var chars = {
            '│' : '|',
            '└' : '`',
            '├' : '+',
            '─' : '-',
            '┬' : '-'
        };
        return opts.unicode === false ? chars[s] : s;
    };

    if (typeof obj === 'string') obj = { label : obj };

    var nodes = obj.nodes || [];
    var lines = (obj.label || '').split('\n');
    var splitter = '\n' + prefix + (nodes.length ? chr('│') : ' ') + ' ';

    return prefix
        + lines.join(splitter) + '\n'
        + nodes.map(function (node, ix) {
            var last = ix === nodes.length - 1;
            var more = node.nodes && node.nodes.length;
            var prefix_ = prefix + (last ? ' ' : chr('│')) + ' ';

            return prefix
                + (last ? chr('└') : chr('├')) + chr('─')
                + (more ? chr('┬') : chr('─')) + ' '
                + archy(node, prefix_, opts).slice(prefix.length + 2)
            ;
        }).join('')
    ;
};
```

The state of the registry

- 1400MM package downloads last month
- 150,000 packages in the registry
- `Node` is moving faster than `Node.js`

Module-driven development

- npm focused on module-driven development
- ES6 Modules
- Module-driven development for the browser
  - Browser tools: Browserify

Module-driven development for the browser

```
bower install moment --save
npm install moment --save
```

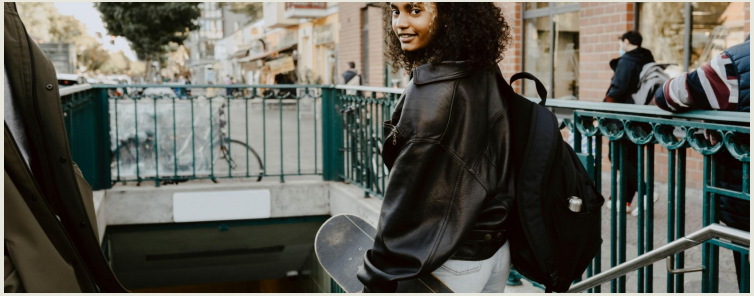
DEMO

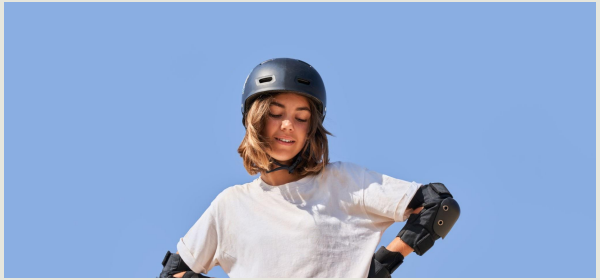
Where can learn more?

- http://nodeschool.io
- http://www.codewars.com
- https://www.codeschool.com/courses/real-time-web-with-node-js
- http://nodeup.com/

HOW TO JOIN

THANK





YOU