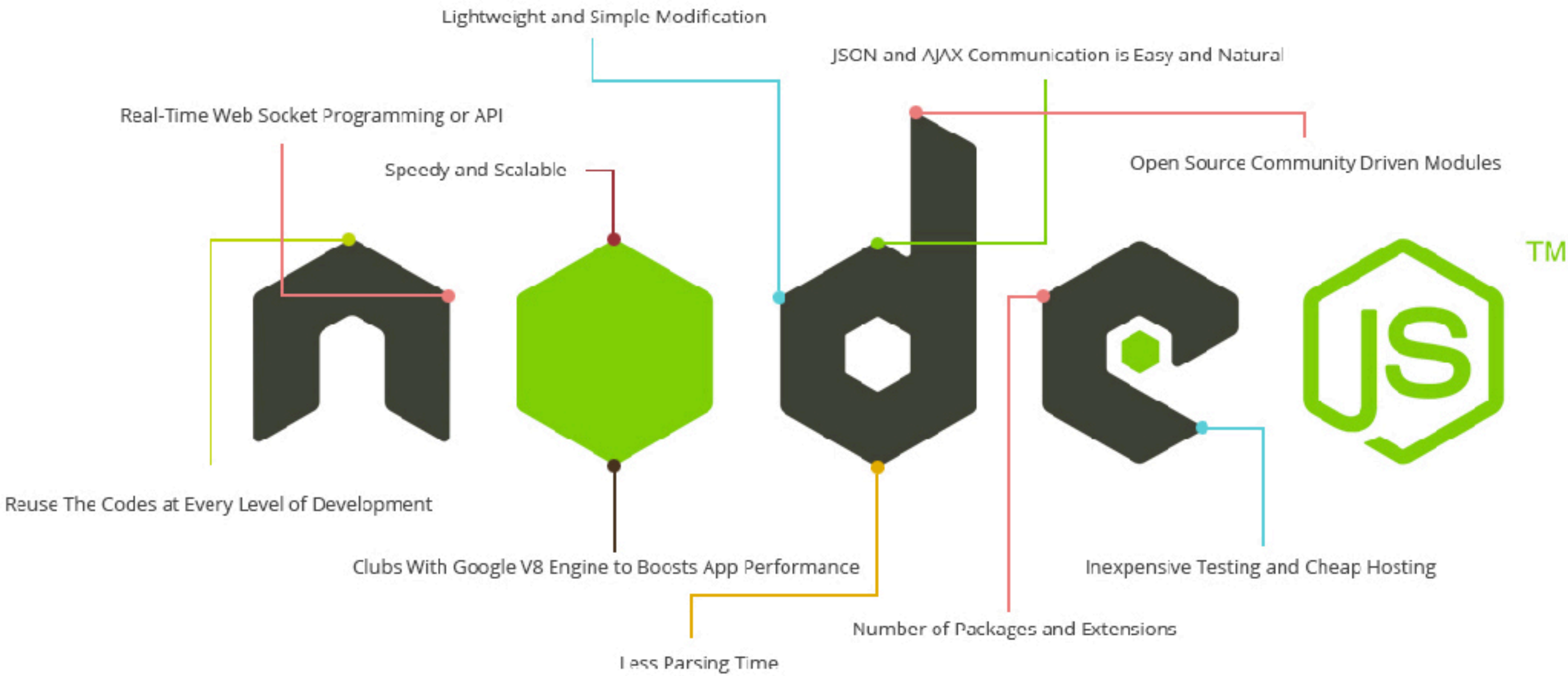
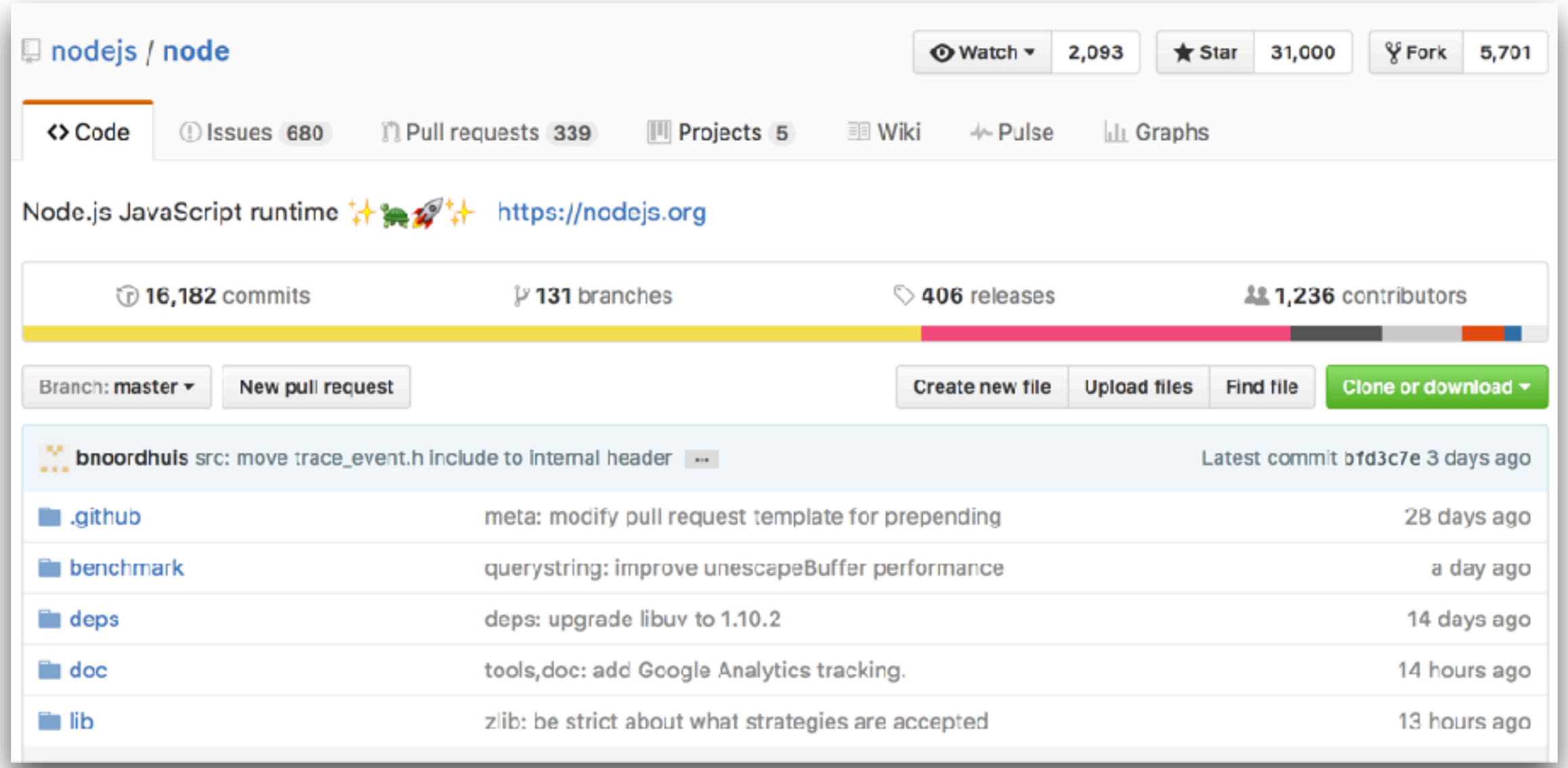
in action 5 days

@somkiat.cc


Lightweight and Simple Modification
JSON and AJAX Communication is Easy and Natural
Real-Time Web Socket Programming or API
Speedy and Scalable
Open Source Community Driven Modules
TM
Reuse The Codes at Every Level of Development
Clubs With Google V8 Engine to Boosts App Performance
Inexpensive Testing and Cheap Hosting
Number of Packages and Extensions
Less Parsing Time

Popular ?

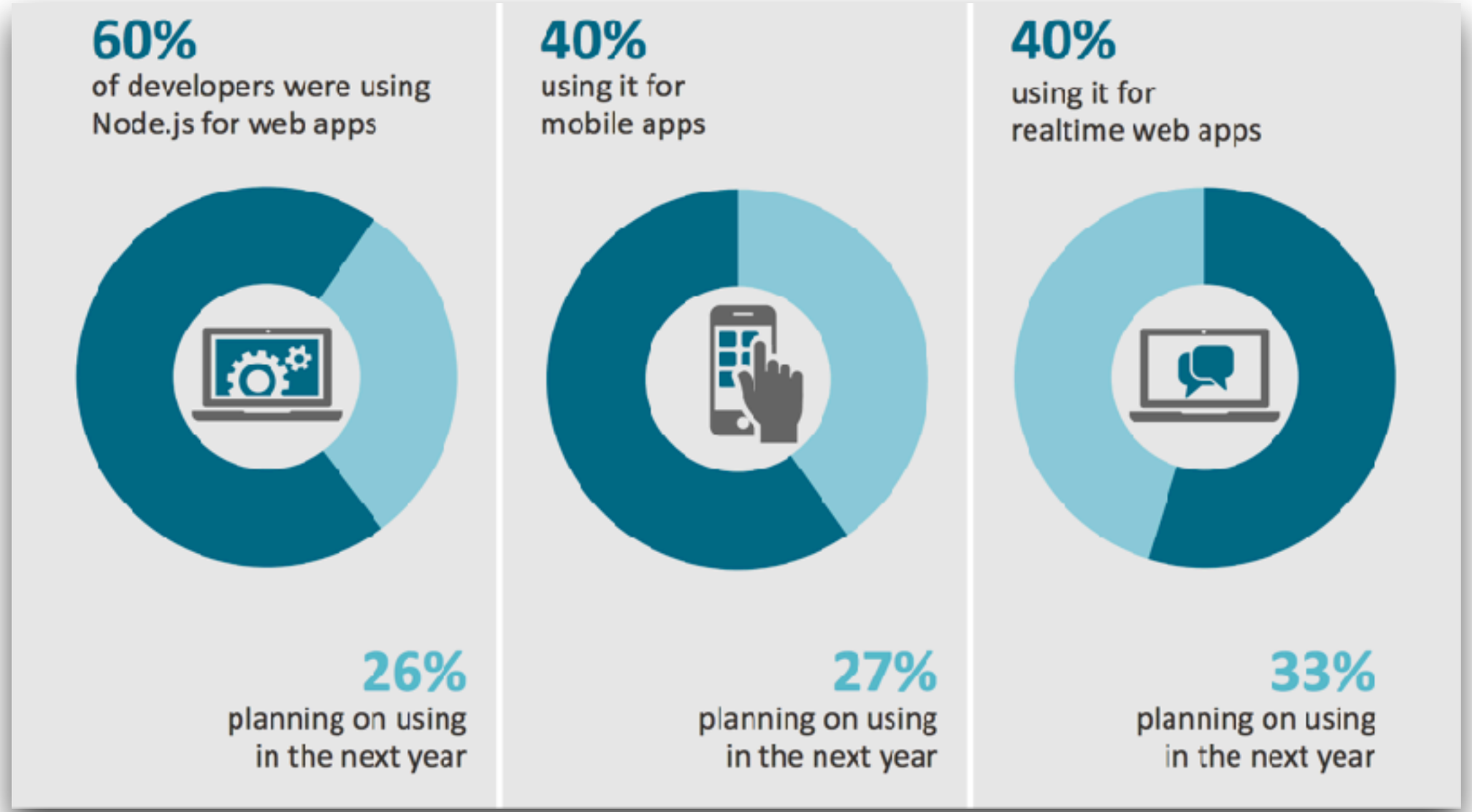


https://github.com/nodejs/node

Who use Node.js ?

https://strongloop.com/node-js/why-node/

Trend of Node.js



https://strongloop.com/node-js/why-node/

Day 1

Welcome to Node.js

What is Node.js ?

JavaScript on server

Asynchronous and Event loop

Type of application with Node.js

Sample app

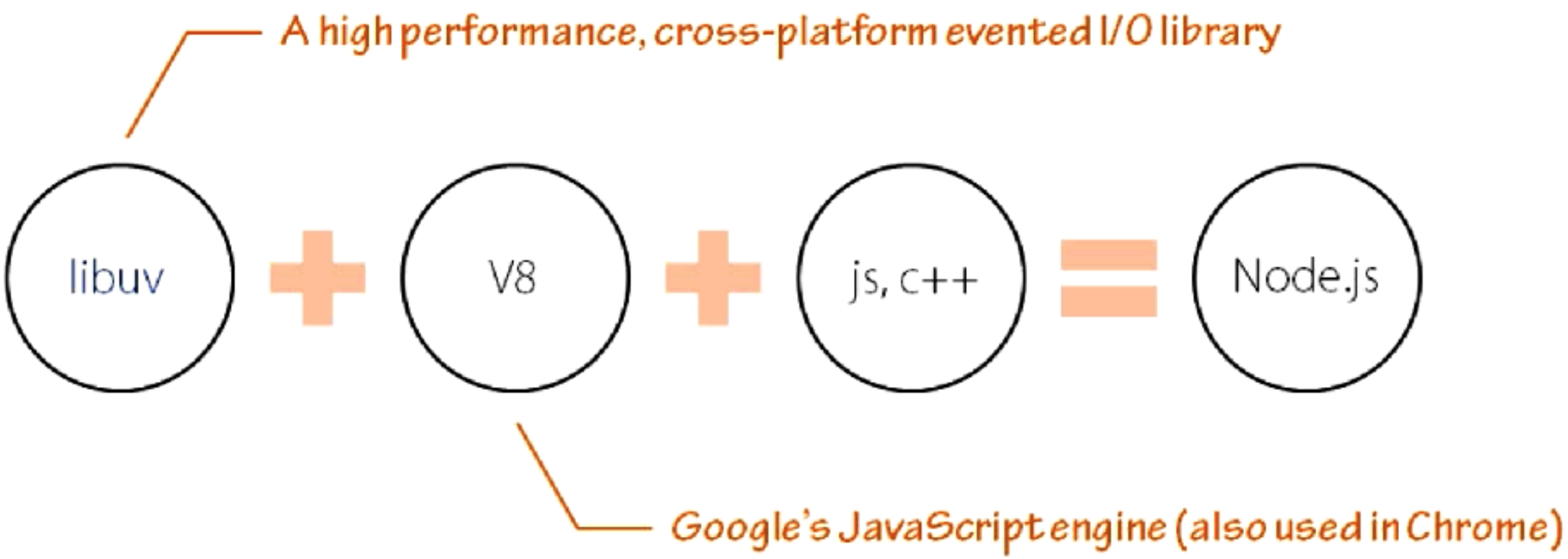
What is Node.js ?

Platform

build on Chrome's **JavaScript** runtime

Fast and **Scalable** application

Building block



What is Node.js ?

Event-driven ?

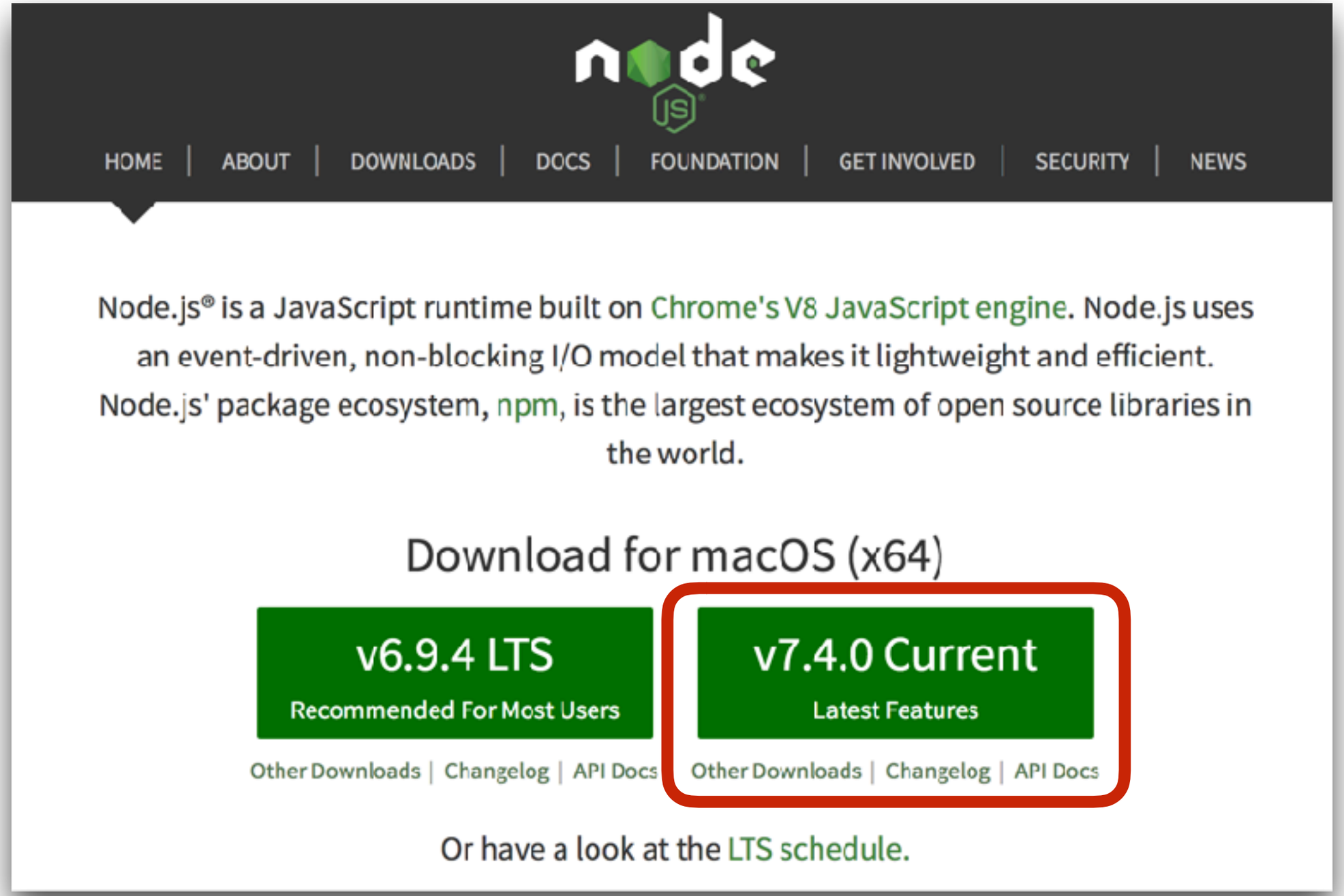
Non-blocking I/O ?

What is Node.js ?

Event-driven ?

Non-blocking I/O ?

Installation



Hello Node.js

$node -v

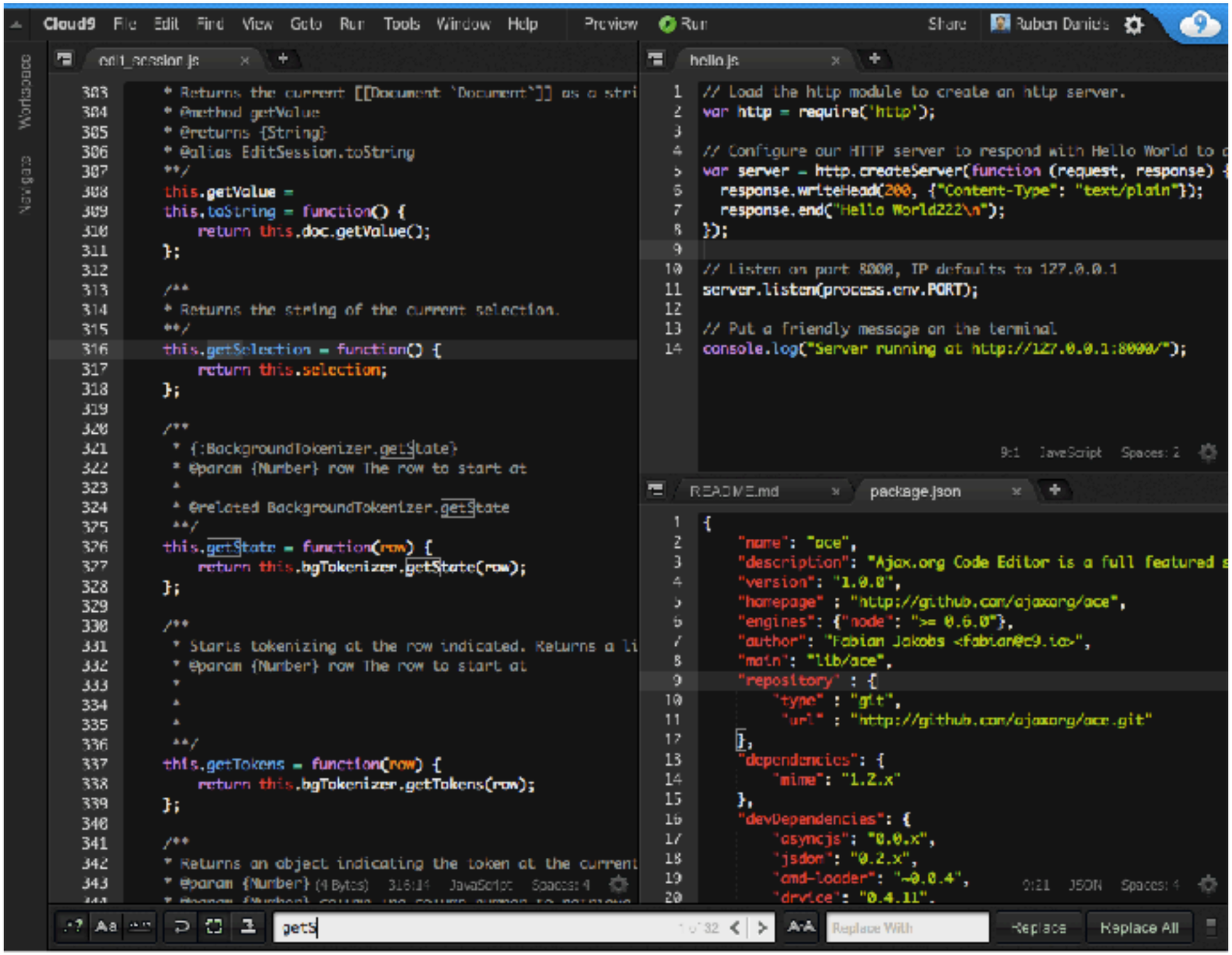
Develop with IDE

Atom

Sublime

Cloud9 IDE

C9.io



Build on JavaScript

Language Types (click to hide)

Web | Mobile | Enterprise | Embedded

Language Rank	Types	Spectrum Ranking
1. Java	Web, Mobile, Enterprise	98.1
2. Python	Web, Enterprise	98.0
3. C#	Web, Mobile, Enterprise	86.7
4. PHP	Web	82.8
5. JavaScript	Web, Mobile	82.2
6. Ruby	Web, Enterprise	74.5
7. Go	Web, Enterprise	71.9
8. Scala	Web, Mobile	66.9
9. HTML	Web	65.6
10. Perl	Web, Enterprise	58.5

Build on JavaScript

Write once, Run anywhere

Frontend and Backend

Node use V8

Virtual machine by google chrome

Boost performance

https://developers.google.com/v8/

Benefits

Write web app in **one language**

Working with **JSON** (native)

Working with **NoSQL** database

Follow **ECMAScript** standard

Asynchronous & Event driven

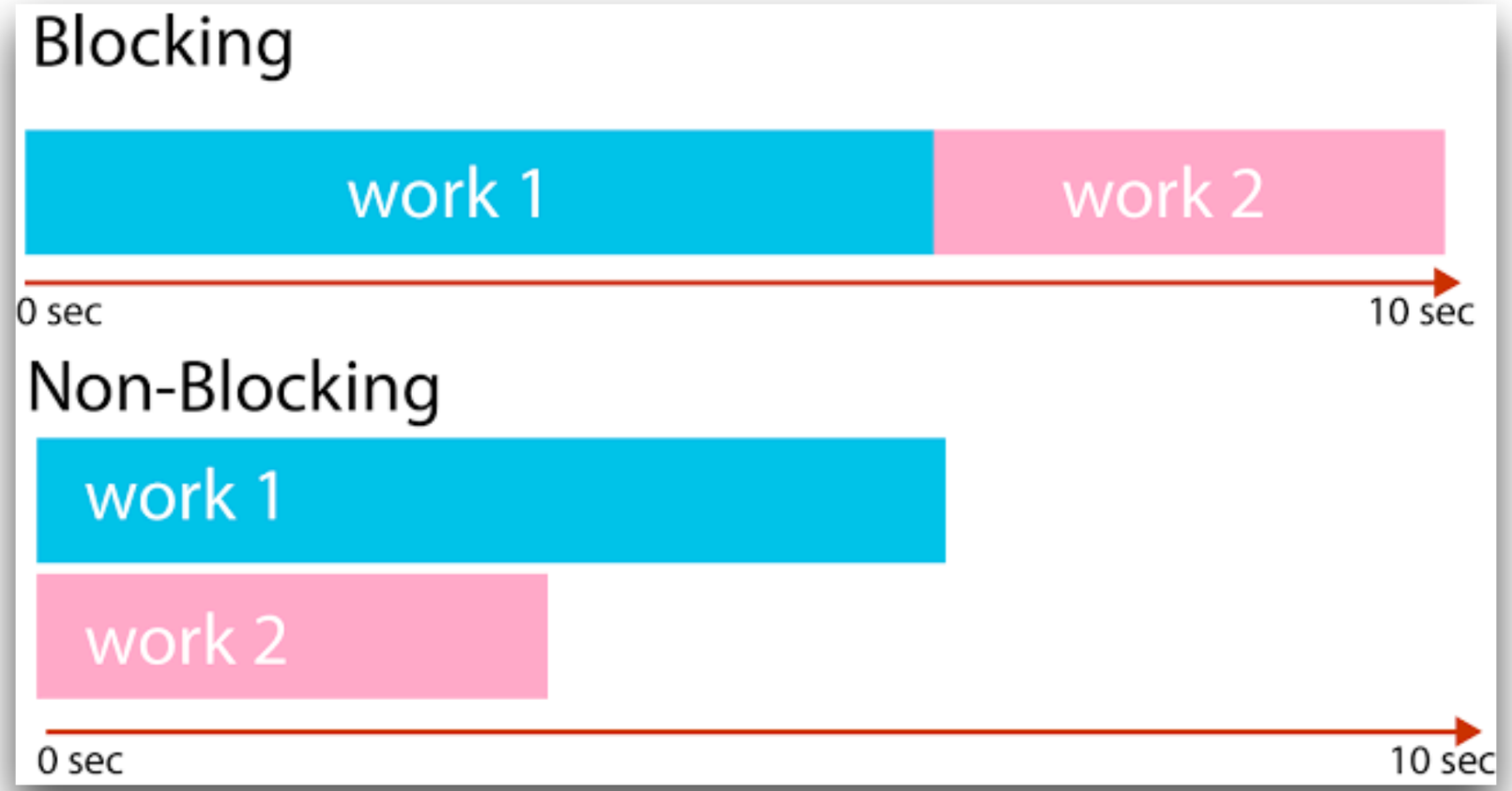
Non-blocking I/O

Event loop

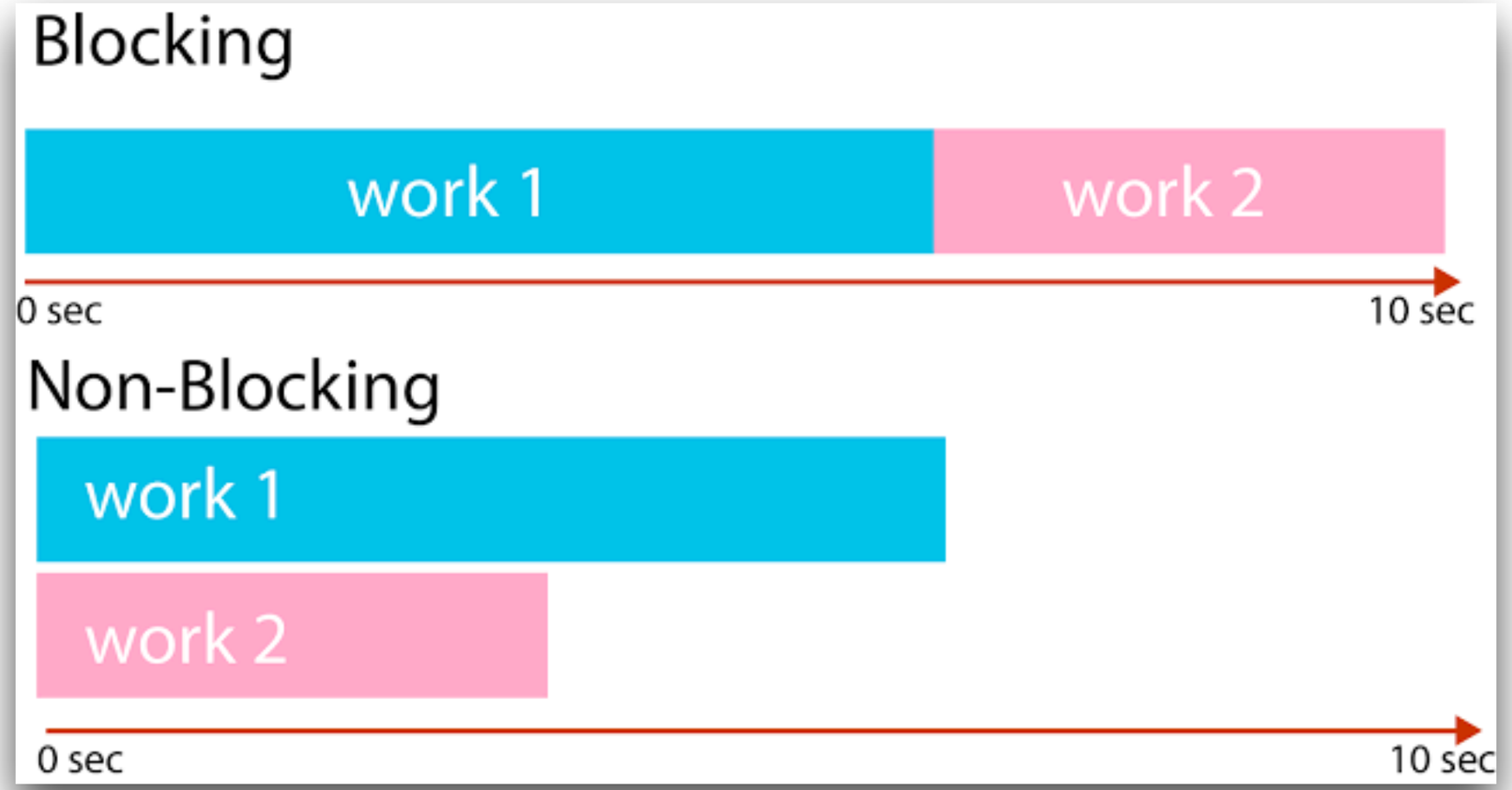
Blocking vs Non-blocking I/O

"I/O" refers primarily to interaction with
the system's **disk** and **network** supported

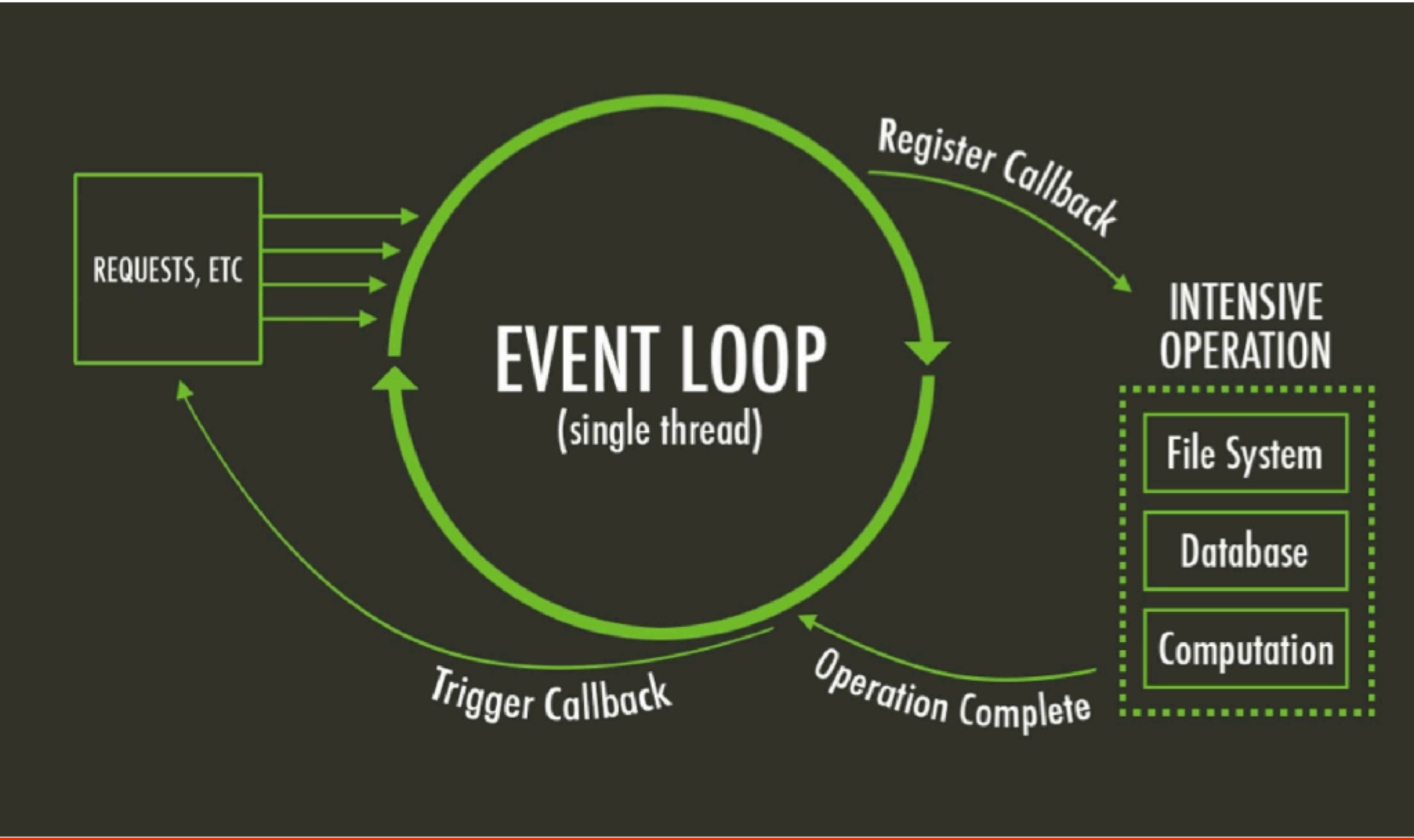
Blocking vs Non-blocking I/O



Synchronous vs Asynchronous



Event loop


REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE
OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete

Non-blocking by default

All of the I/O method in standard library provide

asynchronous (non-blocking)

Blocking I/O

Method name is end with **Sync**

eg.

readFile**Sync**()

Let’s coding

Basic of JavaScript

Basic of JavaScript

Variables and Assignment

Operators

Conditionals

Loops

Functions

Variables and Assignment

```
var counter = 0;
var message = "Hello world";
var isLogin = false;

var users = ["User 1", "User 2"];
var mixData = [1, 2, false, "User 1", "User 2"];
```

Variables and Assignment

```
var dataDictionary = {"id": 1, "name": "Name 1"};


console.log(dataDictionary['id']);
console.log(dataDictionary.id);
console.log(dataDictionary.name);
```

Variables and Assignment

```
var userList = [{"id": 1, "name": "Name 1"},
                {"id": 2, "name": "Name 2"}];


console.log(userList[0].id);
console.log(userList[0].name);
```

Variables and Assignment

```
var add = function(firstNumber, secondNumber){
    return firstNumber + secondNumber;
}


console.log(add(1, 2));
```

Loop

```
var users = ["User 1", "User 2"];


for (var i = 0; i < users.length; i++) {
   console.log(users[i]);
}
```

Loop

```
var users = ["User 1", "User 2"];

for (var index in users) {
   console.log(index + "=>" + users[index]);
}
```

Loop

```
var users = ["User 1", "User 2"];


for (user of users) {
   console.log(user);
}
```

Loop

```
var users = ["User 1", "User 2"];

users.forEach(function(user){
    console.log(user);
})
```

Loop

```
var users = ["User 1", "User 2"];


while(user = users.reverse().pop()){
   console.log(user);
}
```

Functions

```
function hello(name) {
    console.log("Hello " + name);
}

hello("somkiat");
```

Functions

```
var hello = function(name) {
   console.log("Hello " + name);
}

hello("somkiat");
```

Functions

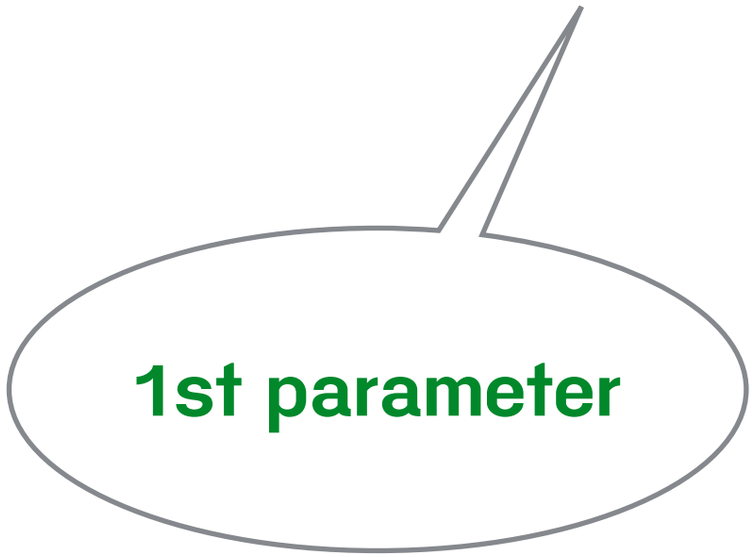
```
var hello = (name) => {
   console.log("Hello " + name)
}

hello("somkiat");
```

Start with Node.js

Send parameter from command line

Let’s start

$node command.js **somkiat**



Try by yourself

```
process.argv.forEach((val, index) => {
  console.log(`${index}: ${val}`);
});
```

https://nodejs.org/docs/latest/api/process.html#process_process_argv

Blocking I/O

Read data from file system

```
const fs = require('fs');
const data = fs.readFileSync('./data.json');
```

Blocking I/O

$node sync_read_file.js

```
Start read file ...
{
  "result": "Hello world!!"
}

Finish read file
```

Non-blocking I/O

Read data from file system

```
const fs = require('fs');
fs.readFile('./data.json', (error, data) => {
    if(error) throw error
    console.log(data.toString());
});
```

Non-blocking I/O

Callback function

```
const fs = require('fs');
fs.readFile('./data.json', (error, data) => {
    if(error) throw error
    console.log(data.toString());
});
```

Non-blocking I/O

Callback function

```
const fs = require('fs');
fs.readFile('./data.json', function(error, data){
    if(error) throw error
    console.log(data.toString());
});
```

Non-blocking I/O

Callback function

```
const fs = require('fs');
var callback = function(error, data) {
  if(error) throw error
  console.log(data.toString());
};
fs.readFile('./data.json', callback);
```

Non-blocking I/O

$node async_read_file.js

```
Start read file ...
Finish read file
{
  "result": "Hello world!!"
}
```

Try by yourself

```
fs.readFile('./data.json', callback);
fs.readFile('./data2.json', callback);
```

Try by yourself

```
setTimeout( function(error, content{
    console.log("Call ...");
}, 2000);

console.log("Finish");
```

Mixing blocking and non-blocking

Bad pattern

```
const fs = require('fs');
fs.readFile('data.json', (err, data) => {
  if (err) throw err;
  console.log(data);
});
fs.unlinkSync('data.json');
```

Bad pattern

```
const fs = require('fs');
fs.readFile('data.json', (err, data) => {
  if (err) throw err;
  console.log(data);
});
fs.unlinkSync('data.json');
```

Non-blocking

Blocking

Good pattern

Non-blocking

```
const fs = require(‘fs');
fs.readFile('data.json', (err, data) => {
    if (err) throw err;
    console.log(data);

    fs.unlink('data.json', (err) => {
        if (err) throw err;
    });
});
```

Non-blocking

Callback Hell !!

The pyramid of doom ...

node95.js

```
1  var floppy = require('floppy');
2
3  floppy.load('disk1', function (data1) {
4      floppy.prompt('Please insert disk 2', function () {
5          floppy.load('disk2', function (data2) {
6              floppy.prompt('Please insert disk 3', function () {
7                  floppy.load('disk3', function (data3) {
8                      floppy.prompt('Please insert disk 4', function () {
9                          floppy.load('disk4', function (data4) {
10                             floppy.prompt('Please insert disk 5', function () {
11                                 floppy.load('disk5', function (data5) {
12                                     // if node.js would have existed in 1995
13                                 });
14                             });
15                         });
16                     });
17                 });
18             });
19         });
20     });
21 });
22
```

Hello HTTP Server

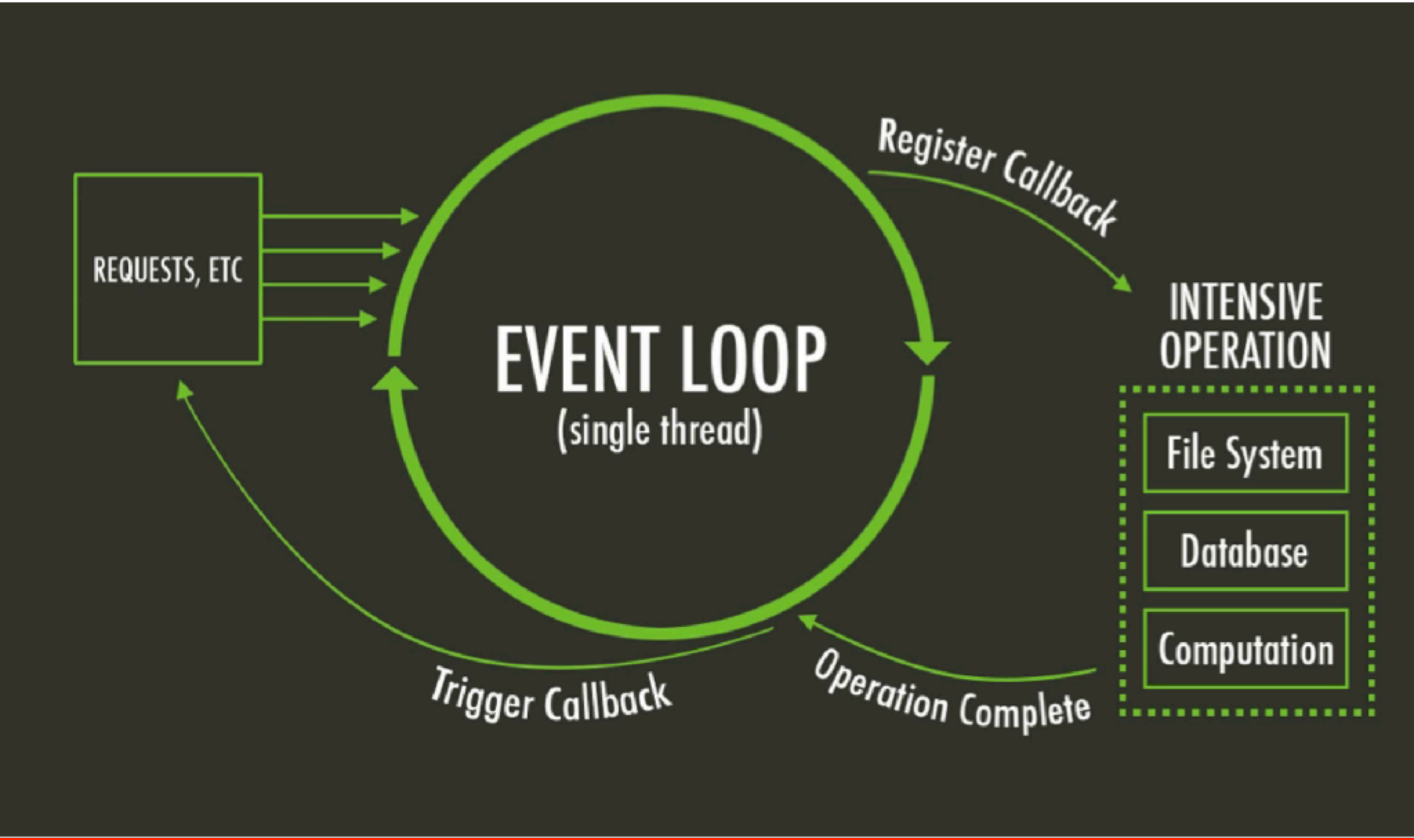
Hello HTTP Server

```
var http = require('http');
http.createServer(function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/plain'});
   res.end('Hello World\n');
}).listen(3000);

console.log('Server running at http://localhost:3000/');
```

Hello HTTP Server

$node hello_server.js

Event loop

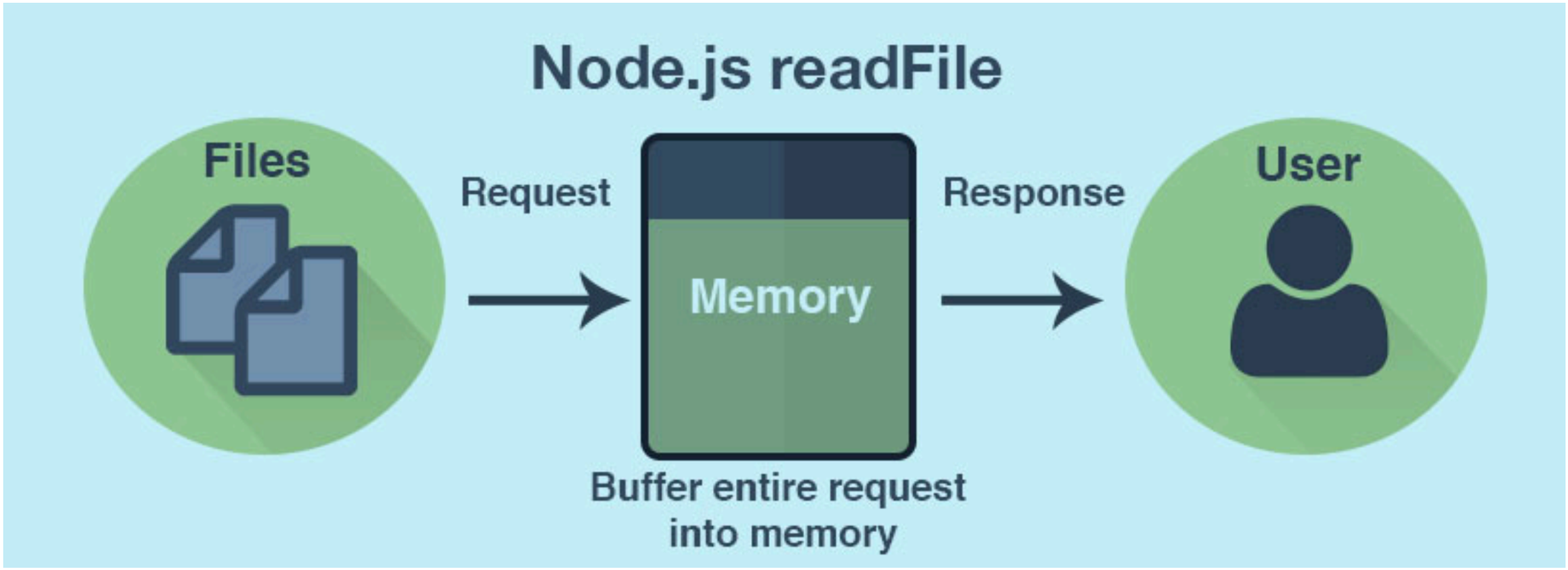

REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE
OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete

Improve structure

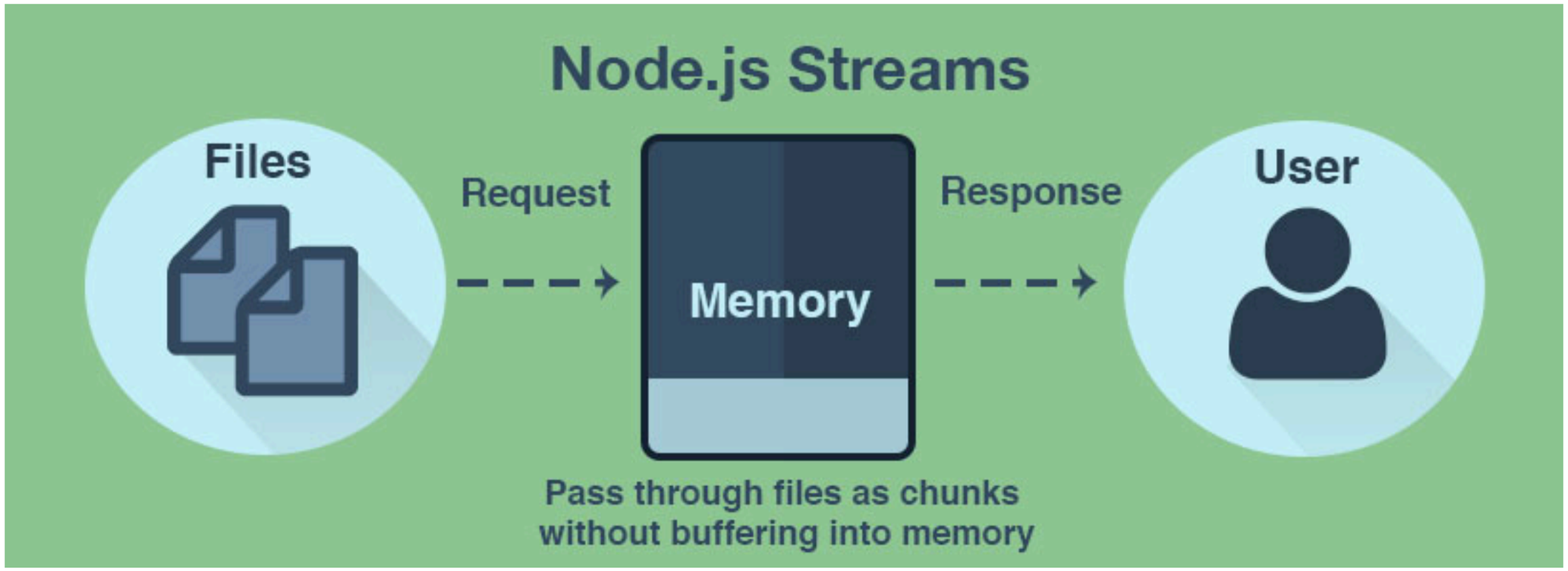
```
var http = require('http');

var server = http.createServer();
server.on('request', function(request, response){
   response.writeHead(200, {'Content-Type': ‘text/plain’});
   response.end('Hello World\n');
});
server.listen(3000);

console.log('Server running at http://localhost:3000/');
```

Streaming data server

Read file with Blocking



Read file with stream



Read file with stream

```
var fs = require('fs');
var stream = fs.createReadStream('data.txt', 'UTF-8');
var data = '';
```

Read file with stream

```
stream.once('data', function(){
   console.log('Started Reading File');
});

stream.on('data', function(chunk){
   process.stdout.write(`chunk: ${chunk.length} \n`);
   data += chunk;
});

stream.on('end', function(){
  console.log(`Finished Reading File ${data.length}`);
});
```

Develop streaming server

```
var http = require('http');
var fs = require('fs');

http.createServer(function(request, response){
   response.writeHead(200, {'Content-Type': 'application/pdf'});
   var stream = fs.createReadStream('data.txt', 'UTF-8');
   stream.pipe(response);
}).listen(3000);

console.log('Server running at http://localhost:3000/');
```

Workshop with Upload file ?

Upload file

```
http.createServer(function(request, response){
    var stream = fs.createWriteStream('data2.txt', 'UTF-8');
    request.pipe(stream);

  request.on('end', function() {
     response.end('uploaded!');
  });

}).listen(3000);
```

Create file on server

Upload file

```
http.createServer(function(request, response){
    var stream = fs.createWriteStream('data2.txt', 'UTF-8');
    request.pipe(stream);

    request.on('end', function() {
        response.end('uploaded!');
    });

}).listen(3000);
```

Handle end event

Summary

Build on JavaScript

Event and Asynchronous

Designed for data-intensive app

Designed for real-time app

Day 2

Building web with Node.js

Building web with Node.js

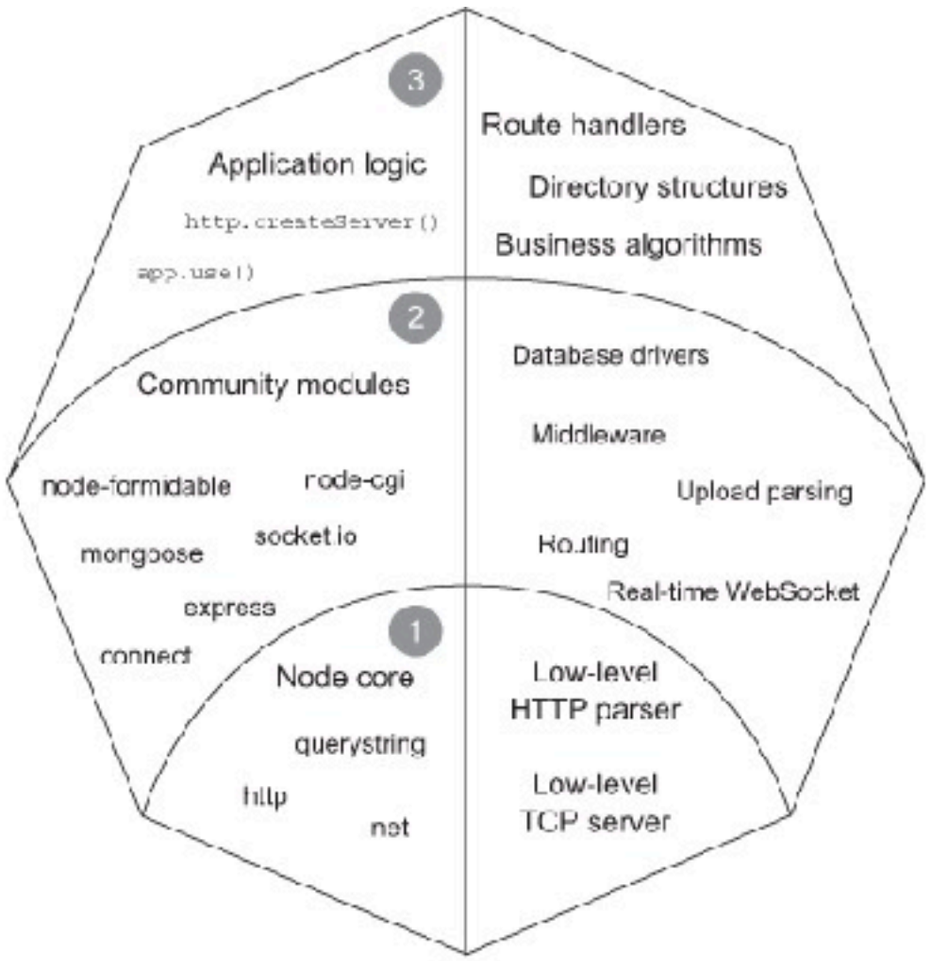
Handling HTTP requests

Building RESTful APIs

Serving static files

Accept user input from forms

Layer of Node web app



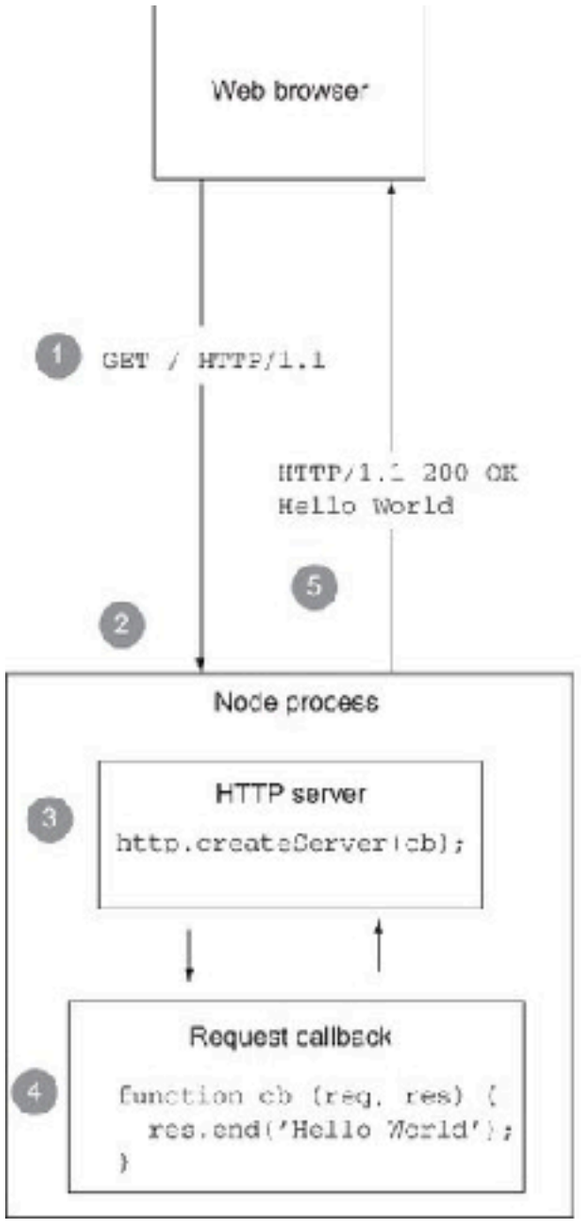
1. Node's core APIs are always lightweight and low-level. This leaves opinions, syntactic sugar, and specific details up to the community modules.

2. Community modules are where Node thrives. Community members take the low-level core APIs and create fun and easy-to-use modules that allow you to get tasks done easily.

3. The application logic layer is where your app is implemented. The size of this layer depends on the number of community modules used and the complexity of the application.

Basic of HTTP server

HTTP Server



1. An HTTP client, like a web browser, initiates an HTTP request.
2. Node accepts the connection, and incoming request data is given to the HTTP server.
3. The HTTP server parses up to the end of the HTTP headers and then hands control over to the request callback.
4. The request callback performs application logic, in this case responding immediately with the text "Hello World."
5. The request is sent back through the HTTP server, which formats a proper HTTP response for the client.

Handling request

```
var http = require('http');
var server = http.createServer(function(req, res){
   // handle request
});
```

Send response

```
var http = require('http');
var server = http.createServer(function(req, res){
   res.write('Hello World');
   res.end();
});
```

Send response

```
var http = require('http');
var server = http.createServer(function(req, res){
   res.end('Hello World');
});
```

Listen for incoming request

```
var http = require('http');
var server = http.createServer(function(req, res){
   res.end('Hello World');
});
server.listen(3000);
```

Setting response headers

```
var http = require('http');
var server = http.createServer(function(req, res){
   var body = 'Hello World';
   res.setHeader('Content-Length', body.length);
   res.setHeader('Content-Type', 'text/plain');
   res.statusCode = 200;
   res.end(body);
});
server.listen(3000);
```

Building

RESTFul ?

- GET
- POST
- PUT
- DELETE

RESTFul

POST : Add data

GET : Retrieve data

DELETE : Delete data

PUT : Update data

Let’s coding

express

http://expressjs.com/

Express

Fast

Easy and Simple

Flexible

Minimal

Installation

$npm install express

Hello express

```
var express = require('express');
var app = express();


app.get('/', function(request, response){
  response.setHeader('Content-Type', 'text/plain');
  response.end('Hello, world!');
});

app.listen(3000);
```

Routing

Routing

```
app.get('/', function(request, response){
  response.end('Hello world with GET');
});

app.post('/', function(request, response){
   response.end('Hello world with POST');
});
```

Routing with parameters

```
app.get('/users/:userId/books/:bookId',
        function (req, res) {

  res.send(req.params)

})
```

App.route()

```
app.route('/book')
   .get(function (req, res) {
       res.send('Get a random book')
   })

 .post(function (req, res) {
       res.send('Add a book')
 })

.put(function (req, res) {
       res.send('Update the book')
 });
```

Route modular

```
var express = require('express')
var router = express.Router()


router.get('/', function (req, res) {
   res.send('Birds home page')
})

router.get('/about', function (req, res) {
   res.send('About birds')
})

module.exports = router
```

Route modular

```
var birds = require(‘./birds');
app.use('/birds', birds)
```

Working with JSON

Working with JSON

```
var photos = [];

photos.push({
  name: "First",
  path: "image01.jpg"
});

app.get('/photo', function(request, response){
   response.json(photos)
});
```

Let’s workshop with REST

Let’s workshop with REST

HTTP Method	URL	Action
GET	/photo	Get all photo
GET	/photo/:id	Get photo by id
POST	/photo	Add new photo
PUT	/photo	Update photo
DELETE	/photo/:id	Delete photo by id

Easy to start with generator

Express generator

$npm install express-generator -g

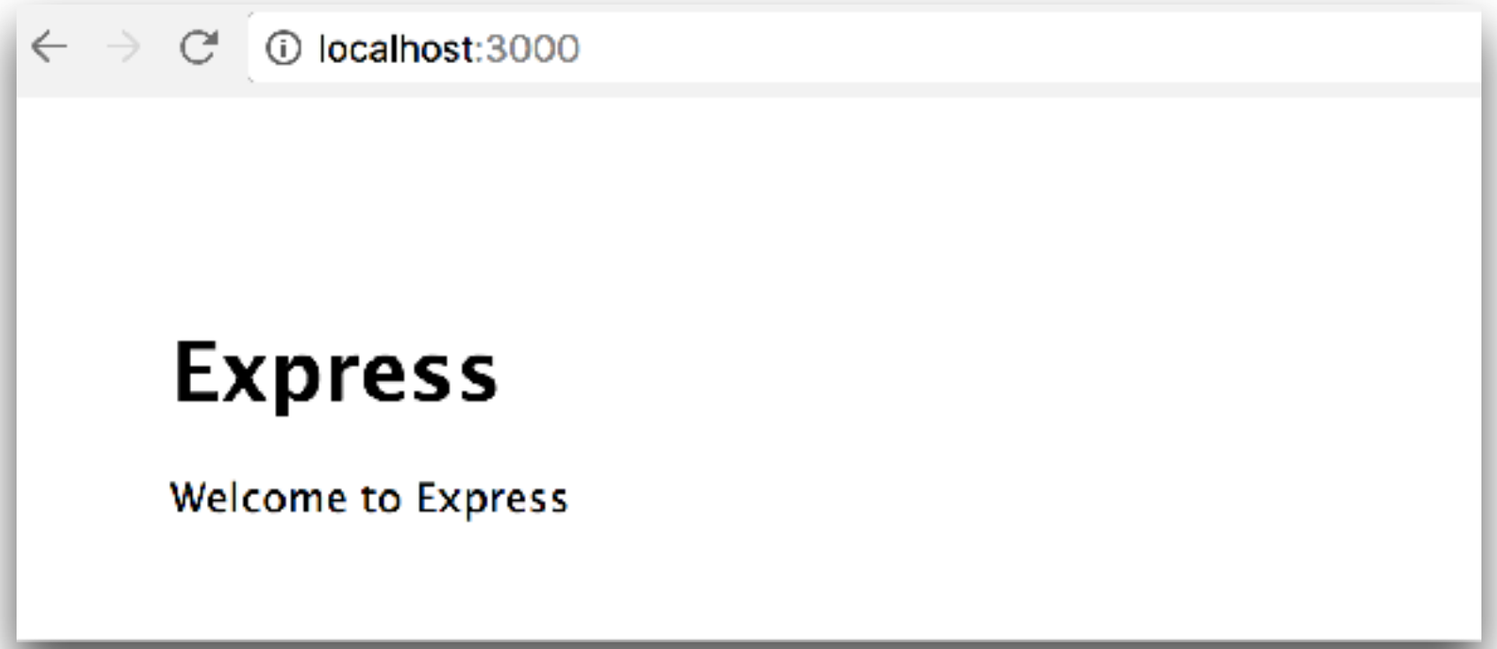
Express generator

$express -h

```
Usage: express [options] [dir]

Options:

  -h, --help          output usage information
      --version       output the version number
  -e, --ejs           add ejs engine support
      --pug           add pug engine support
      --hbs           add handlebars engine support
  -H, --hogan         add hogan.js engine support
  -v, --view <engine> add view <engine> support (ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
  -c, --css <engine>  add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
      --git           add .gitignore
  -f, --force         force on non-empty directory
```

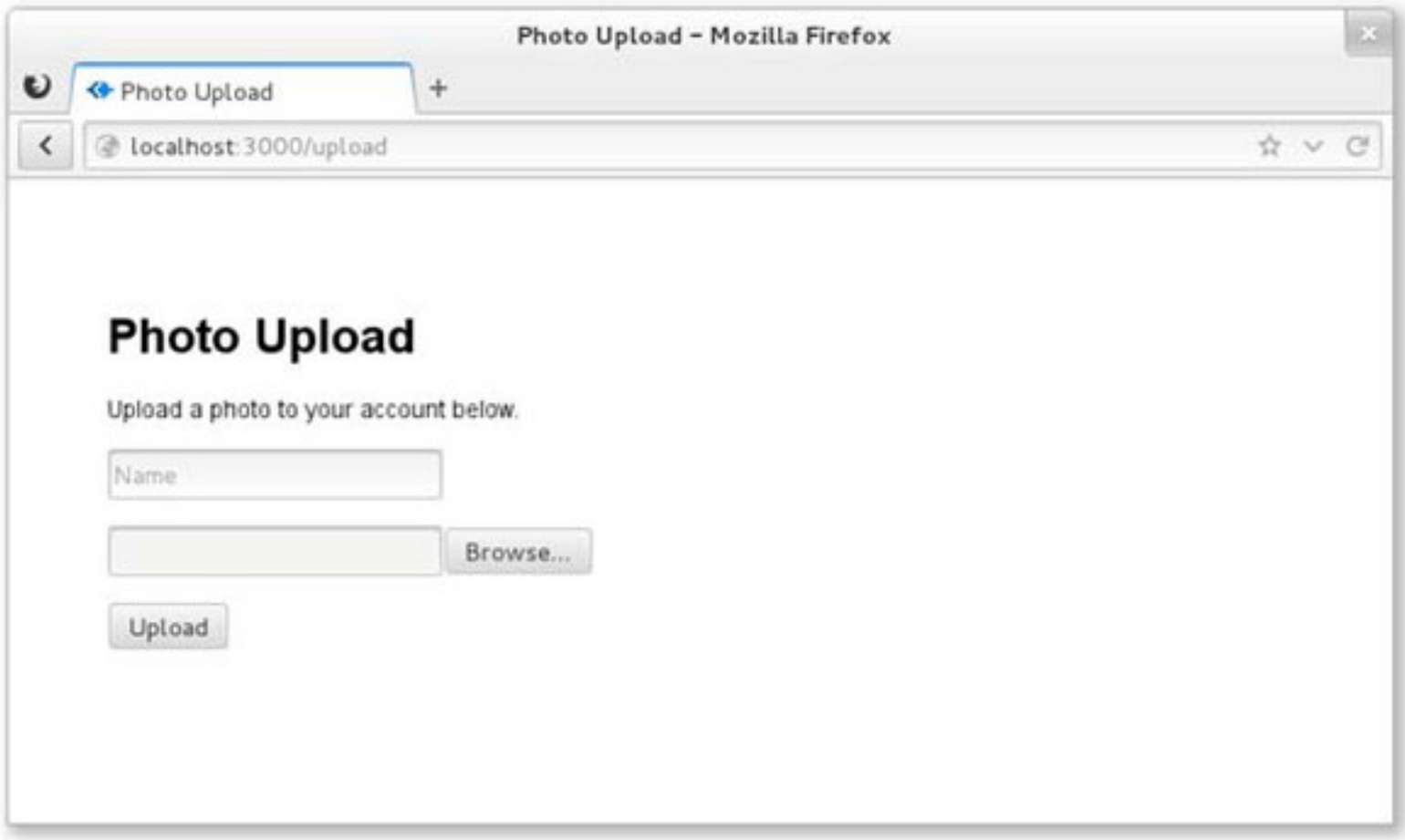
Create application

```
$express --view=ejs myapp
$npm install
$npm start
```

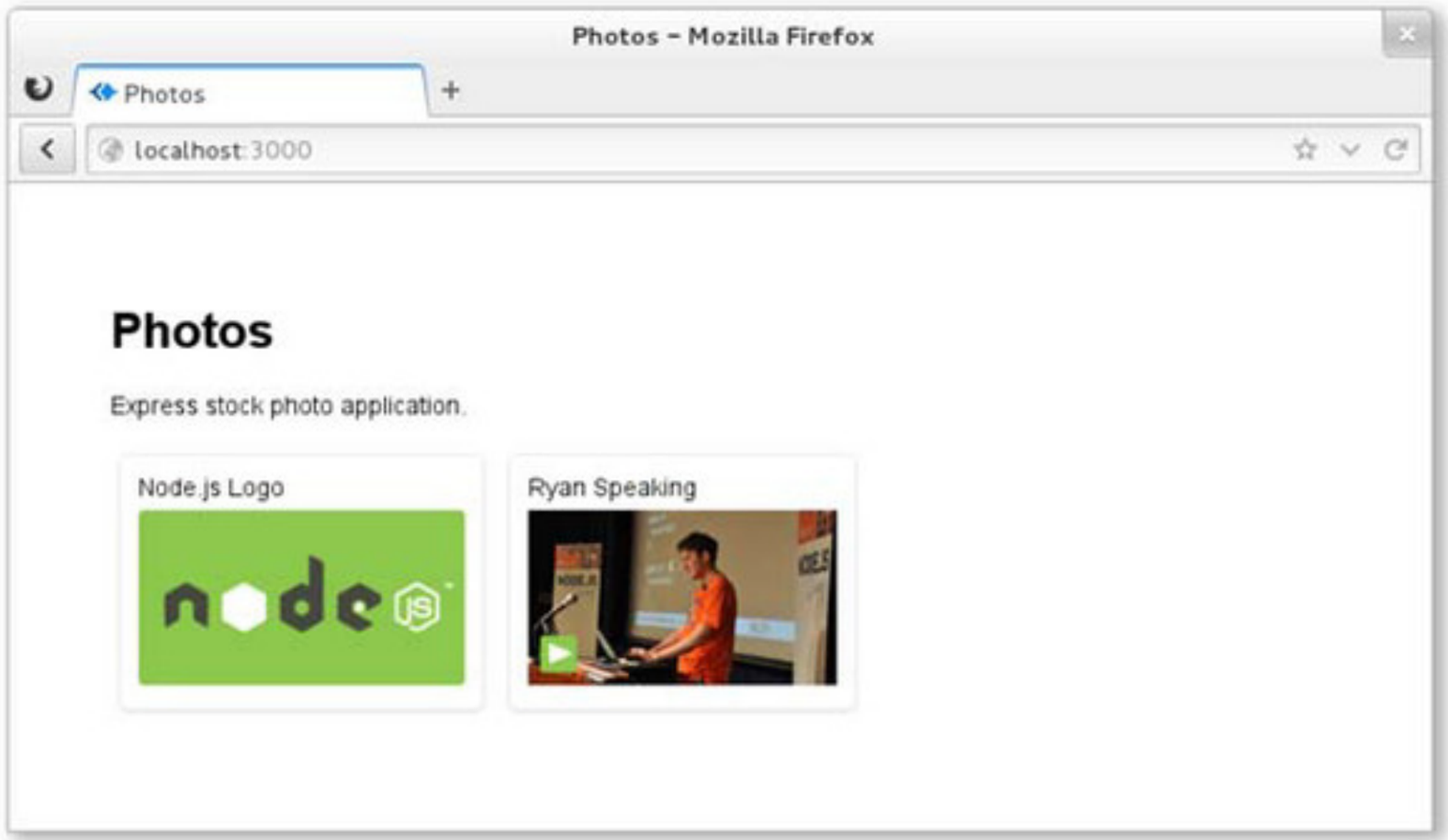


Let’s workshop

Let’s workshop



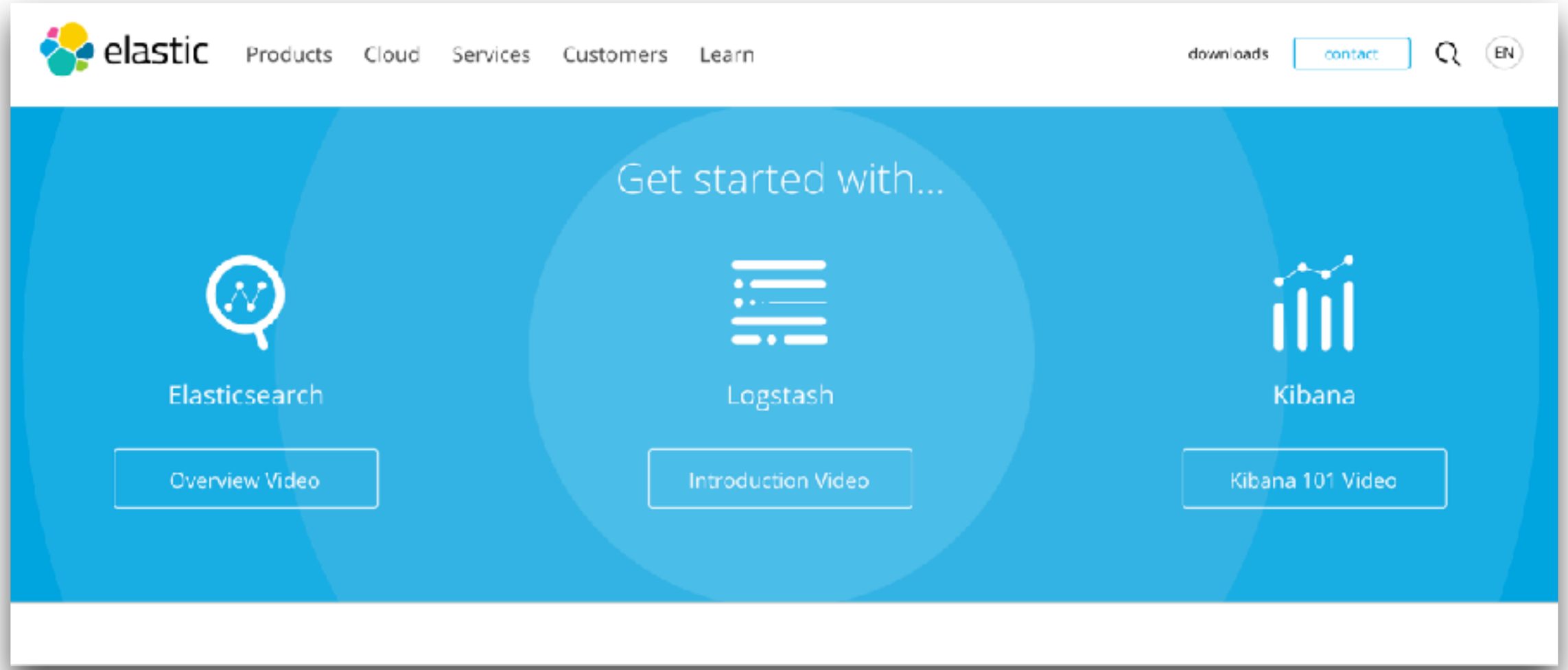
Let’s workshop



Day 3

Manage data

Elasticsearch



https://www.elastic.co/

SELECT

FROM TABLE

WHERE TEXT LIKE ‘%SHIT%’

High-Availability

Plug-ins

Lucene

Distributed

Scalability

RESTFul

JSON

API

elasticsearch

open-source

realtime, search and analytics engine

documentation

document store

JAVA

Use cases

Full-text search

Data store

Analytics

Alert

Ads

Who use ?

Path
github
foursquare
SONY
SOUNDCLOUD
bazaarvoice
XING
salesforce desk
IGN
mozilla
stackoverflow
KLOUT
infochimps
BREWSTER
SCOUT 24
NETWORKED INSIGHTS
Fog Creek SOFTWARE
Revinate
picturesafe

Client libraries

- Java
- PHP
- Ruby
- Python
- JavaScript
- **NodeJS**
- Go
- Scala
- .Net
- Clojure
- Erlang
- R

Let’s start with Elasticsearch

Download and Install

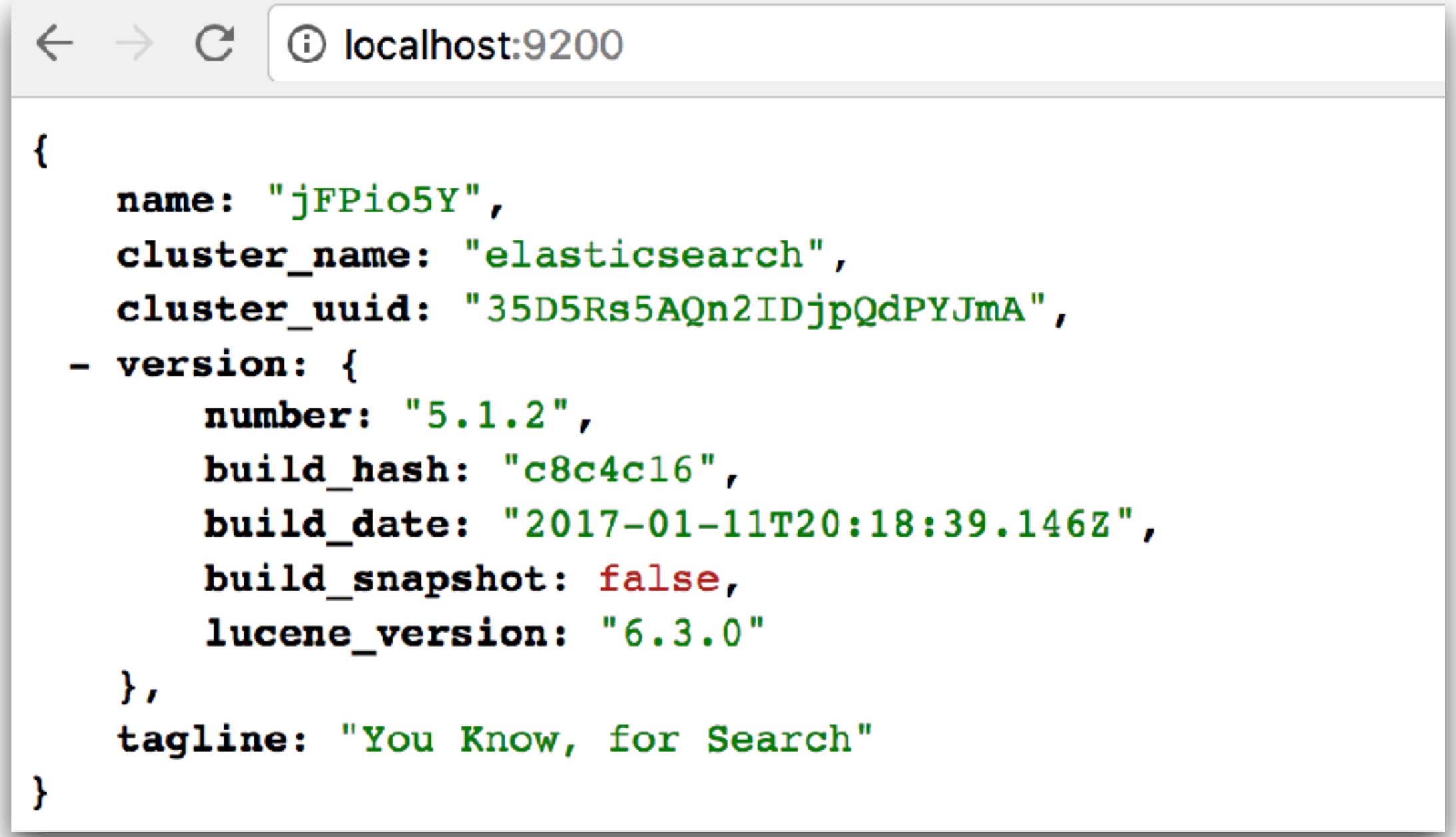
https://www.elastic.co/downloads/elasticsearch

Start elasticsearch

$./bin/elasticsearch

```
[2017-01-27T11:36:29,612][INFO ][o.e.n.Node               ] [] initializing ...
[2017-01-27T11:36:29,746][INFO ][o.e.e.NodeEnvironment    ] [jFPio5Y] using [1] data paths, mounts [[/ (/dev/dis
], net usable_space [32.7gb], net total_space [232.6gb], spins? [unknown], types [hfs]
[2017-01-27T11:36:29,747][INFO ][o.e.e.NodeEnvironment    ] [jFPio5Y] heap size [1.9gb], compressed ordinary obj
pointers [true]
[2017-01-27T11:36:29,749][INFO ][o.e.n.Node               ] node name [jFPio5Y] derived from node ID [jFPio5Y6Rk
iWRhKnqw]; set [node.name] to override
[2017-01-27T11:36:29,754][INFO ][o.e.n.Node               ] version[5.1.2], pid[36610], build[cBc4c16/2017-01-11
18:39.146Z], OS[Mac OS X/10.12.1/x86_64], JVM[Oracle Corporation/Java HotSpot(TM) 64-Bit Server VM/1.8.0_40/25.4
5]
[2017-01-27T11:36:31,825][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [aggs-matrix-stats]
[2017-01-27T11:36:31,825][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [ingest-common]
[2017-01-27T11:36:31,825][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [lang-expression]
[2017-01-27T11:36:31,826][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [lang-groovy]
[2017-01-27T11:36:31,826][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [lang-mustache]
[2017-01-27T11:36:31,826][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [lang-painless]
[2017-01-27T11:36:31,826][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [percolator]
[2017-01-27T11:36:31,826][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [reindex]
[2017-01-27T11:36:31,827][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [transport-netty3]
[2017-01-27T11:36:31,827][INFO ][o.e.p.PluginsService     ] [jFPio5Y] loaded module [transport-netty4]
[2017-01-27T11:36:31,828][INFO ][o.e.p.PluginsService     ] [jFPio5Y] no plugins loaded
[2017-01-27T11:36:36,231][INFO ][o.e.n.Node               ] initialized
[2017-01-27T11:36:36,231][INFO ][o.e.n.Node               ] [jFPio5Y] starting ...
[2017-01-27T11:36:36,604][INFO ][o.e.t.TransportService   ] [jFPio5Y] publish_address {127.0.0.1:9300}, bound_ad
ses {[fe80::1]:9300}, {[::1]:9300}, {127.0.0.1:9300}
[2017-01-27T11:36:39,723][INFO ][o.e.c.s.ClusterService   ] [jFPio5Y] new_master {jFPio5Y}{jFPio5Y6RkaCuEiWRhKnq
```

Hello elasticsearch



Manage data with

Download and Install

https://www.elastic.co/downloads/kibana

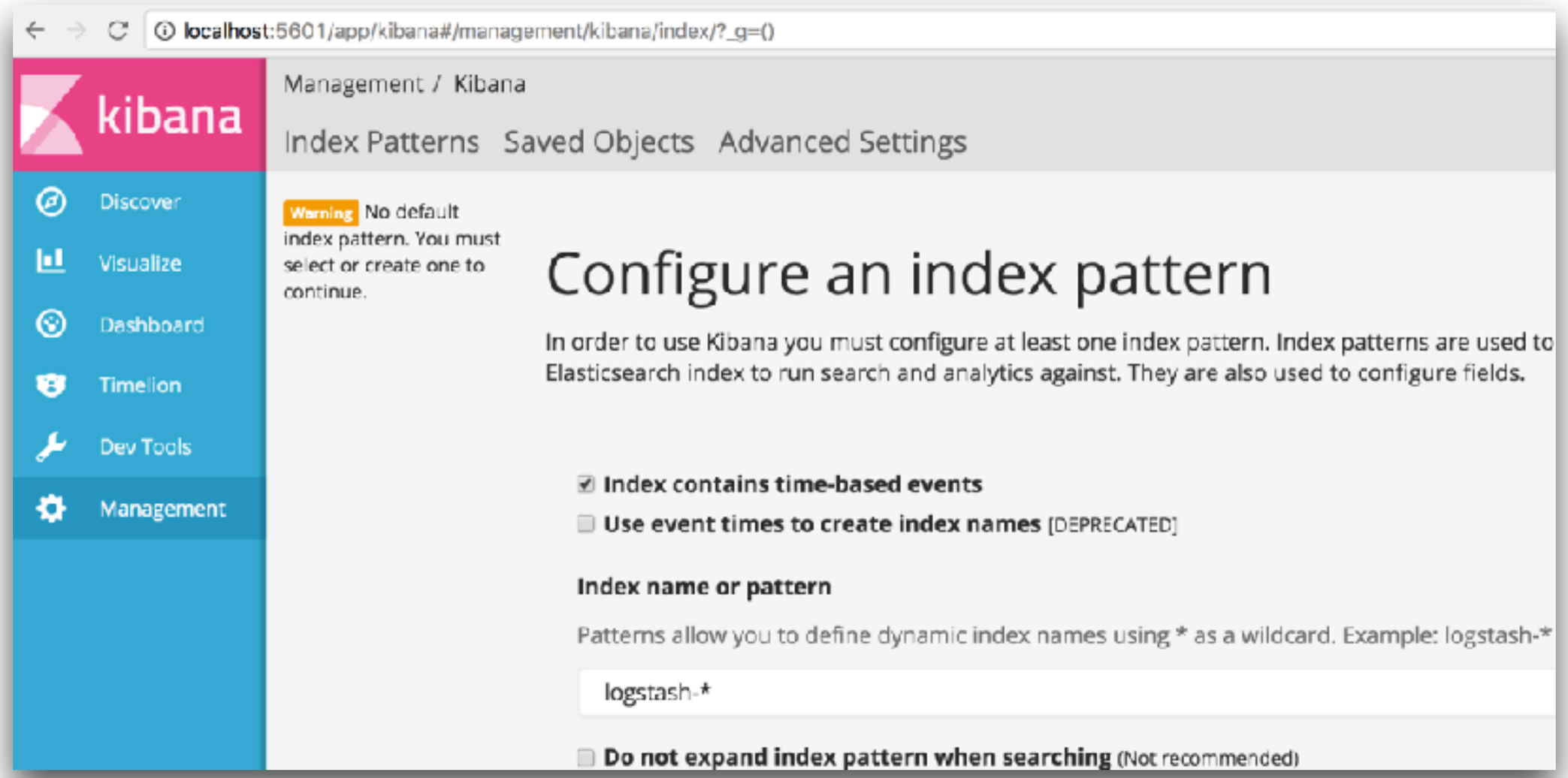
Start kibana

$./bin/kibana

```
  log   [04:44:47.787] [info][status][plugin:kibana@5.1.2] Status changed from uninitialized to
  log   [04:44:47.896] [info][status][plugin:elasticsearch@5.1.2] Status changed from uninitiali
ing for Elasticsearch
  log   [04:44:47.959] [info][status][plugin:console@5.1.2] Status changed from uninitialized to
  log   [04:44:48.345] [info][status][plugin:timelion@5.1.2] Status changed from uninitialized t
  log   [04:44:48.350] [info][listening] Server running at http://localhost:5601
  log   [04:44:48.352] [info][status][ui settings] Status changed from uninitialized to yellow -
n is yellow
  log   [04:44:53.522] [info][status][plugin:elasticsearch@5.1.2] Status changed from yellow to
 Kibana index found
  log   [04:44:54.268] [info][status][plugin:elasticsearch@5.1.2] Status changed from yellow to
 ready
  log   [04:44:54.269] [info][status][ui settings] Status changed from yellow to green - Ready
```

Hello kibana

https://www.elastic.co/downloads/kibana



Dev Tools


kibana
Dev Tools
History
Settings
Help
Console
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
1
1

Manage data

Create

Read

Update

Delete

Compare with RDBMS

DATABASE	TABLE	ROW	COLUMN
INDEX	TYPE	DOCUMENT	FIELD

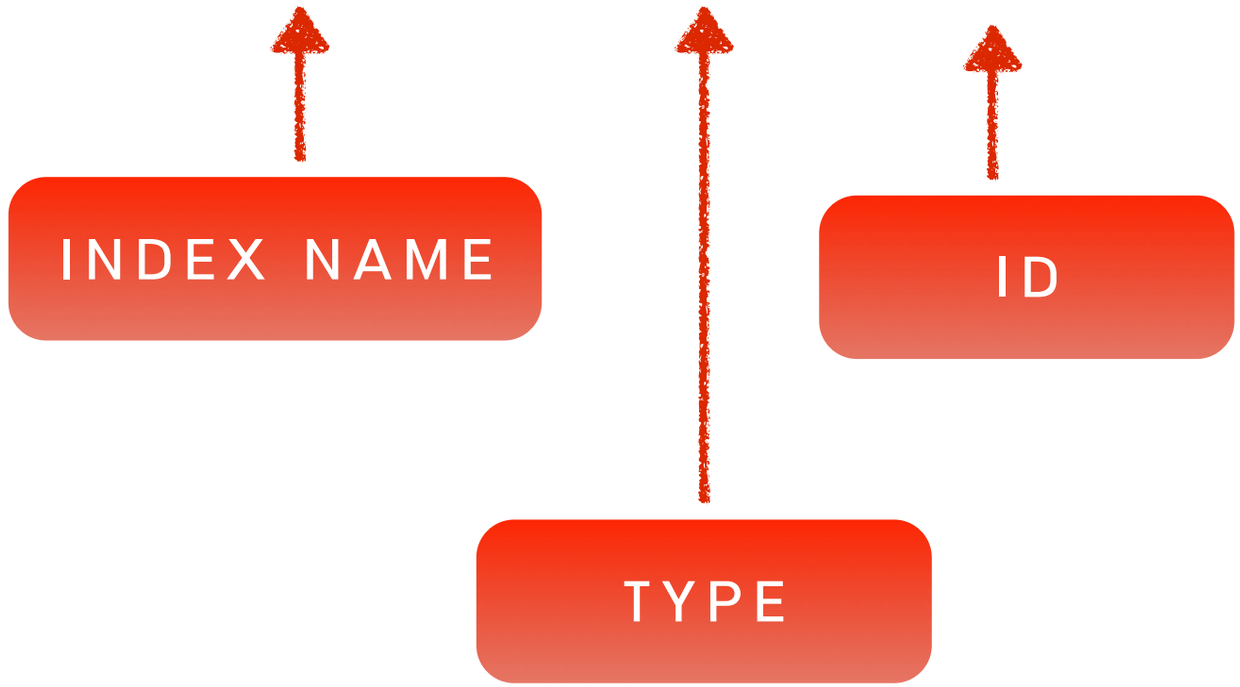
Create data

```
PUT /media/photo/1
{
  "name": "photo 01",
  "path": "/path/image01"
}
```

```
{
  "_index": "media",
  "_type": "photo",
  "_id": "1",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": true
```

Create data

PUT /media/photo/1



Read data

GET /media/photo/1

```
GET /media/photo/1
```

```
{
  "_index": "media",
  "_type": "photo",
  "_id": "1",
  "_version": 1,
  "found": true,
  "_source": {
    "name": "photo 01",
    "path": "/path/image01"
  }
}
```

Update data

PUT /media/photo/1

```
PUT /media/photo/1
{
  "name": "Update photo 01",
  "path": "/path/image01"
}


GET /media/photo/1
```

```
{
  "_index": "media",
  "_type": "photo",
  "_id": "1",
  "_version": 4,
  "result": "updated",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": false
}
```

Update partial data

POST /media/photo/1/**_update**

```
POST /media/photo/1/_update
{
  "doc": {
    "name": "Update partial photo 01"
  }

}


GET /media/photo/1
```

```
{
  "_index": "media",
  "_type": "photo",
  "_id": "1",
  "_version": 9,
  "result": "noop",
  "_shards": {
    "total": 0,
    "successful": 0,
    "failed": 0
  }
}
```

Delete data

DELETE /media/photo/1

```
DELETE /media/photo/1


GET /media/photo/1
```

```
{
  "found": true,
  "_index": "media",
  "_type": "photo",
  "_id": "1",
  "_version": 10,
  "result": "deleted",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  }
}
```

Workshop Query data

Working with Node.js

Install package

$npm install elastic search

Connect to elasticsearch

```
var elasticsearch=require('elasticsearch');


var client = new elasticsearch.Client( {
   hosts: [ 'http://localhost:9200/']
});


client.cluster.health({},function(err,resp,status) {
   console.log("-- Client Health --",resp);
});
```

Connect to elasticsearch

```
var elasticsearch=require('elasticsearch');


var client = new elasticsearch.Client( {
   hosts: [ 'http://localhost:9200/']
});


client.cluster.health({},function(err,resp,status) {
   console.log("-- Client Health --",resp);
});
```

Cluster health

```
var elasticsearch=require('elasticsearch');

var client = new elasticsearch.Client( {
   hosts: [ 'http://localhost:9200/']
});

client.cluster.health({}, function(err,resp,status) {
   console.log("-- Client Health --",resp);
});
```

Cluster health

```
-- Client Health -- { cluster_name: 'elasticsearch',
  status: 'yellow',
  timed_out: false,
  number_of_nodes: 1,
  number_of_data_nodes: 1,
  active_primary_shards: 6,
  active_shards: 6,
  relocating_shards: 0,
  initializing_shards: 0,
  unassigned_shards: 6,
  delayed_unassigned_shards: 0,
  number_of_pending_tasks: 0,
  number_of_in_flight_fetch: 0,
  task_max_waiting_in_queue_millis: 0,
  active_shards_percent_as_number: 50 }
```

Create index

```
client.indices.create({ index: 'media'},
function(err,resp,status) {
   if(err) {
      console.log(err);
   }else {
      console.log("create",resp);
   }

});
```

Delete index

```
client.indices.delete({ index: 'media'},
function(err,resp,status) {
   if(err) {
      console.log(err);
   }else {
      console.log("create",resp);
   }

});
```

Create data

```
client.index({
   index: 'media',
   id: '1',
   type: 'photo',
   body: {
      "name": "image 01",
      "path": "image_01.jpg"
   }
},function(err,resp,status) {
   console.log(resp);
});
```

Create data

```
client.index({
   index: 'media',
   id: '1',
   type: 'photo',
   body: {
      "name": "image 01",
      "path": "image_01.jpg"
   }
},function(err,resp,status) {
   console.log(resp);
});
```

Count data

```
client.count({
    index: 'media',
    type: 'photo'
},
function(err,resp,status) {
        console.log("Photo",resp);
});
```

Delete data

```
client.delete({
    index: 'media',
    id: '1',
    type: 'photo'
},function(err,resp,status) {
    console.log(resp);
});
```

Search data

```
client.search({
    index: 'media',
    type: 'photo',
    body: {
        query: {
            match: { "_all": "image" }
        },
    }
}
```

Search data

```
client.search({
    index: 'media',
    type: 'photo',
    body: {
        query: {
            match: { "_all": "image" }
        },
    }
}
```

Same result

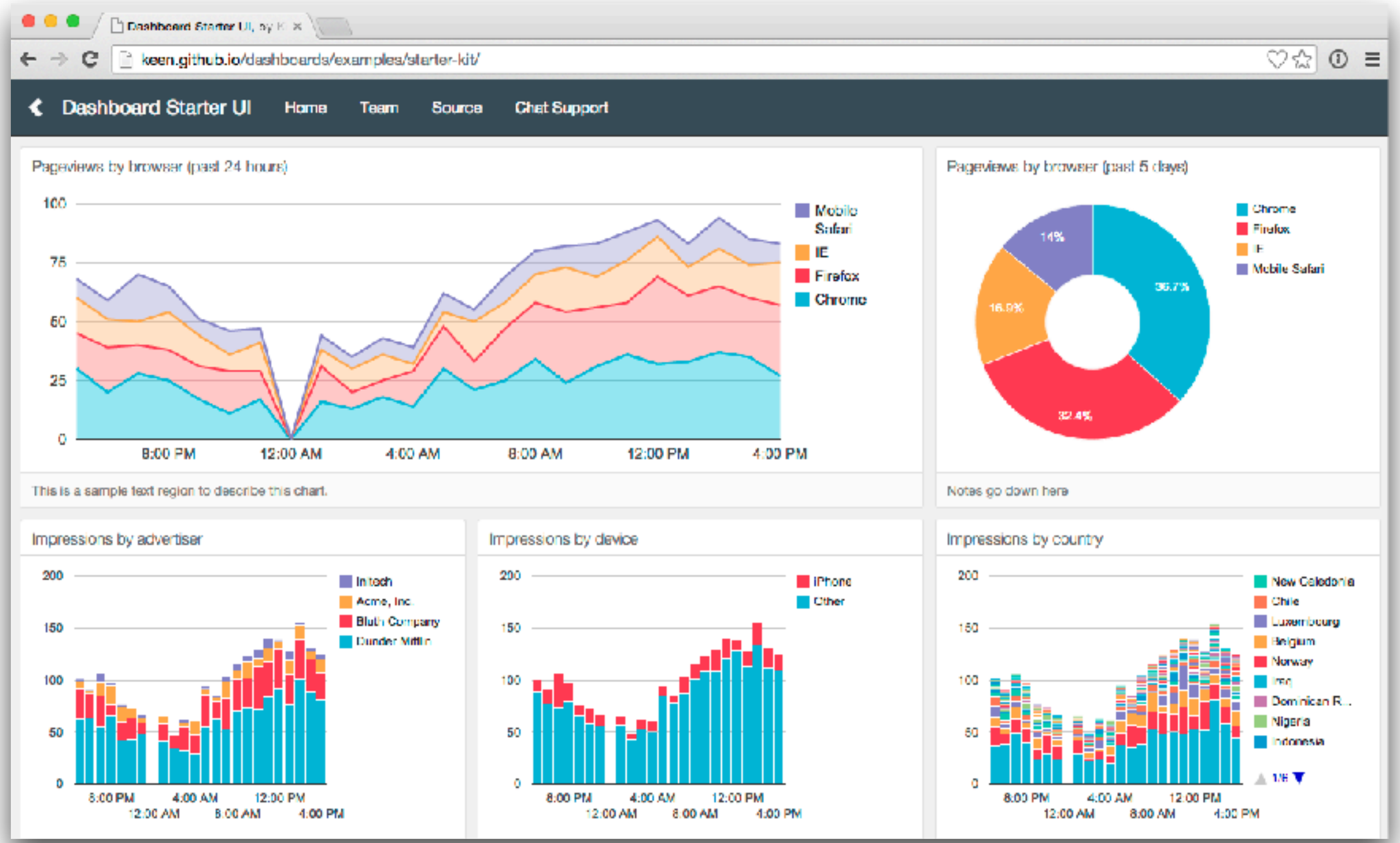
```
GET /media/photo/_search
{
  "query": {
    "match": {
      "_all": "image"
    }
  }
}
```

```
{
  "took": 43,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 0.2876821,
    "hits": [
      {
        "_index": "media",
        "_type": "photo",
        "_id": "1",
        "_score": 0.2876821,
        "_source": {
          "name": "image 01",
          "path": "image_01.jpg"
        }
      }
    ]
  }
}
```

Workshop keep data in Elasticsearch

Day 4 & 5

Realtime analytic



Structure

Workshop