JAVASCRIPT ON THE SERVER. A NODE.JS CRASH COURSE

Group A
Lukas Navickas, Shilpa Ghanashyam Gore, Angelin Rashmi, Tim Henkelmann

AGENDA

1. Node.JS crash course – theoretical background
    1. Node.JS and microprocessors
    2. Node.JS basics
    3. Parts of Node.JS
    4. Tools for Node.JS development
2. Node.JS crash course – live coding
    1. Building a complete conference website with Node.JS
    2. Scraping smashingconf.com website
3. Project roundup – scraping structured data
    1. Our Experiences, remarks, takeaways

AGENDA

1. **Node.JS crash course – theoretical background**
    1. **Node.JS and microprocessors**
    2. Node.JS basics
    3. Parts of Node.JS
    4. Tools for Node.JS development
2. Node.JS crash course – live coding
    1. Building a complete conference website with Node.JS
    2. Scraping smashingconf.com website
3. Project roundup – scraping structured data
    1. Our Experiences, remarks, takeaways

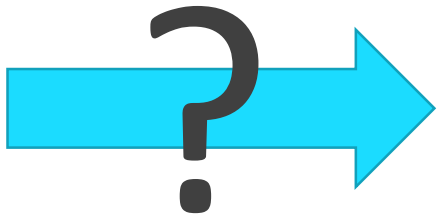
BEFORE DIGGING DEEPER…

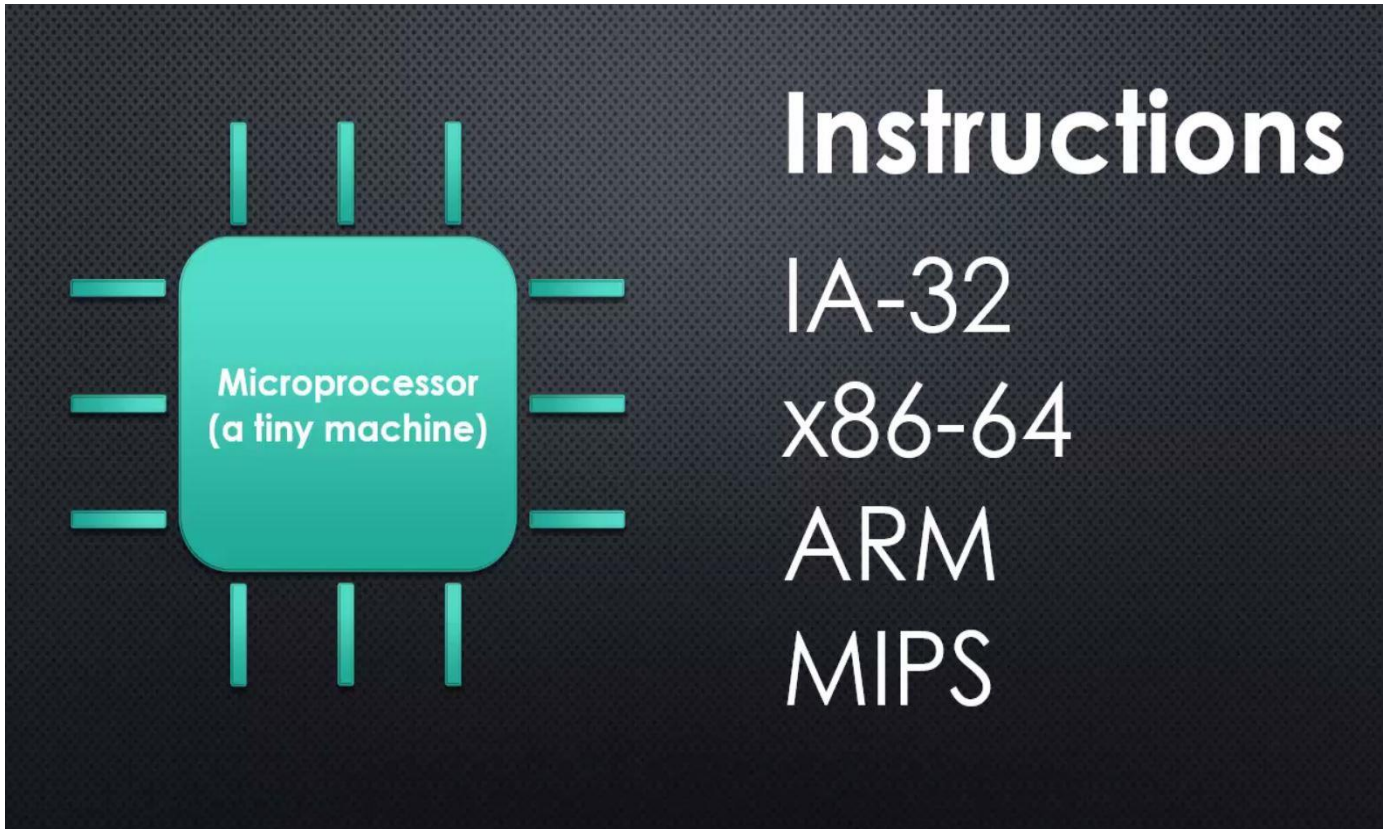
- Processors
- Machine code
- C++

… and Node.JS?

NODE.JS IS CONVERTED IN TO MACHINE CODE?

Node.JS

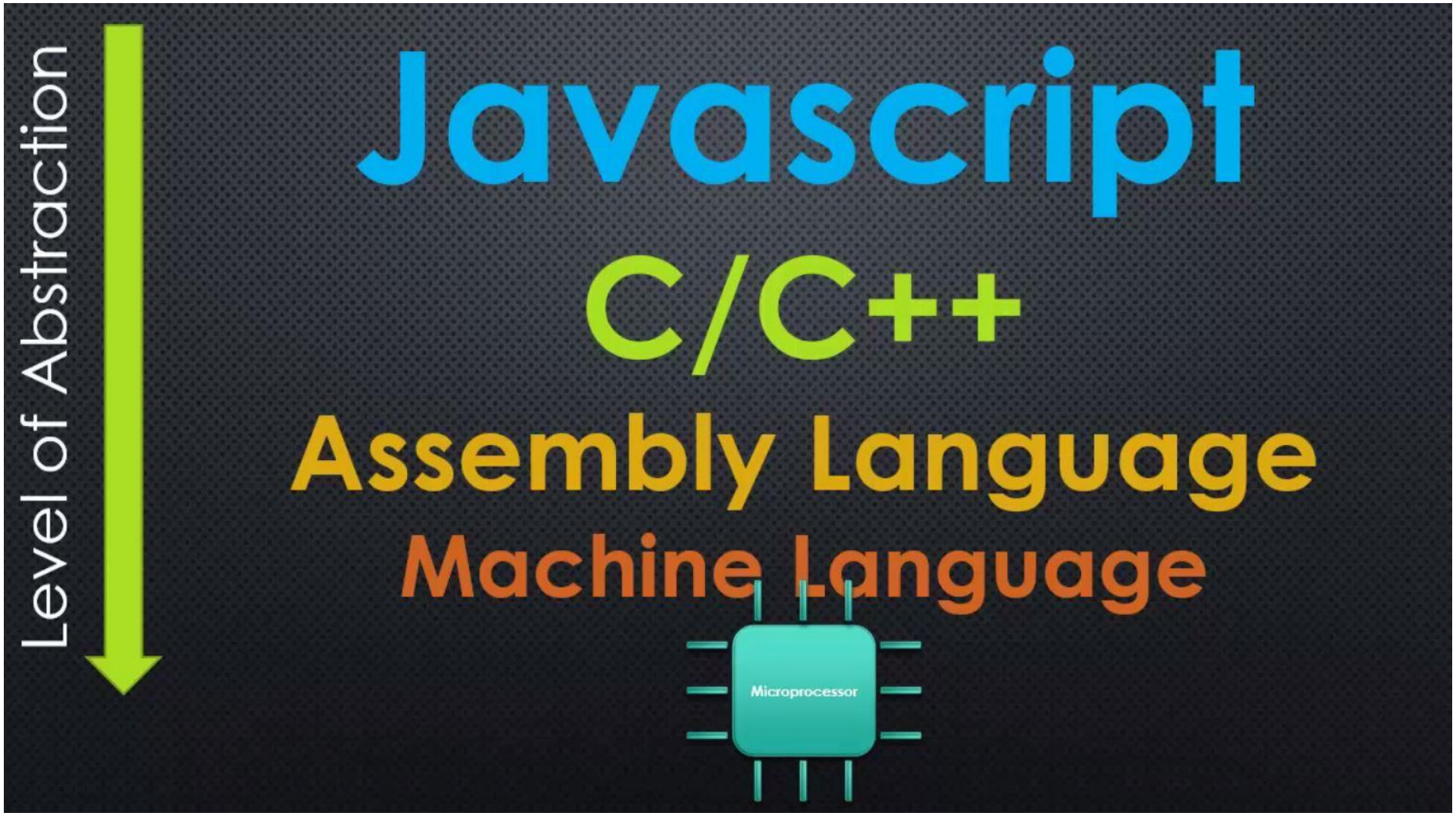




LET’S BUILD A FACEBOOK!

```
000018A45438100      0  55                push rbp
0000018A45438101     1  4889e5            REX.W movq rbp,rsp
0000018A45438104     4  56                push rsi
0000018A45438105     5  57                push rdi
0000018A45438106     6  41ff75a8          push [r13-0x58]
0000018A4543810A    10  56                push rsi
0000018A4543810B    11  49baf9552c7e8f010000 REX.W movq r10,
0000018F7E2C55F9    ;;  object: 0000018F7E2C55F9
0000018A45438115    21  4152              push r10
0000018A45438117    23  6a00              push 0x0
0000018A45438119    25  b803000000        movl rax,000000000000
```

NODE.JS IS NOT HERE ☹



PUBLIC QUESTION #1

In which Language NodeJS is written?

C++

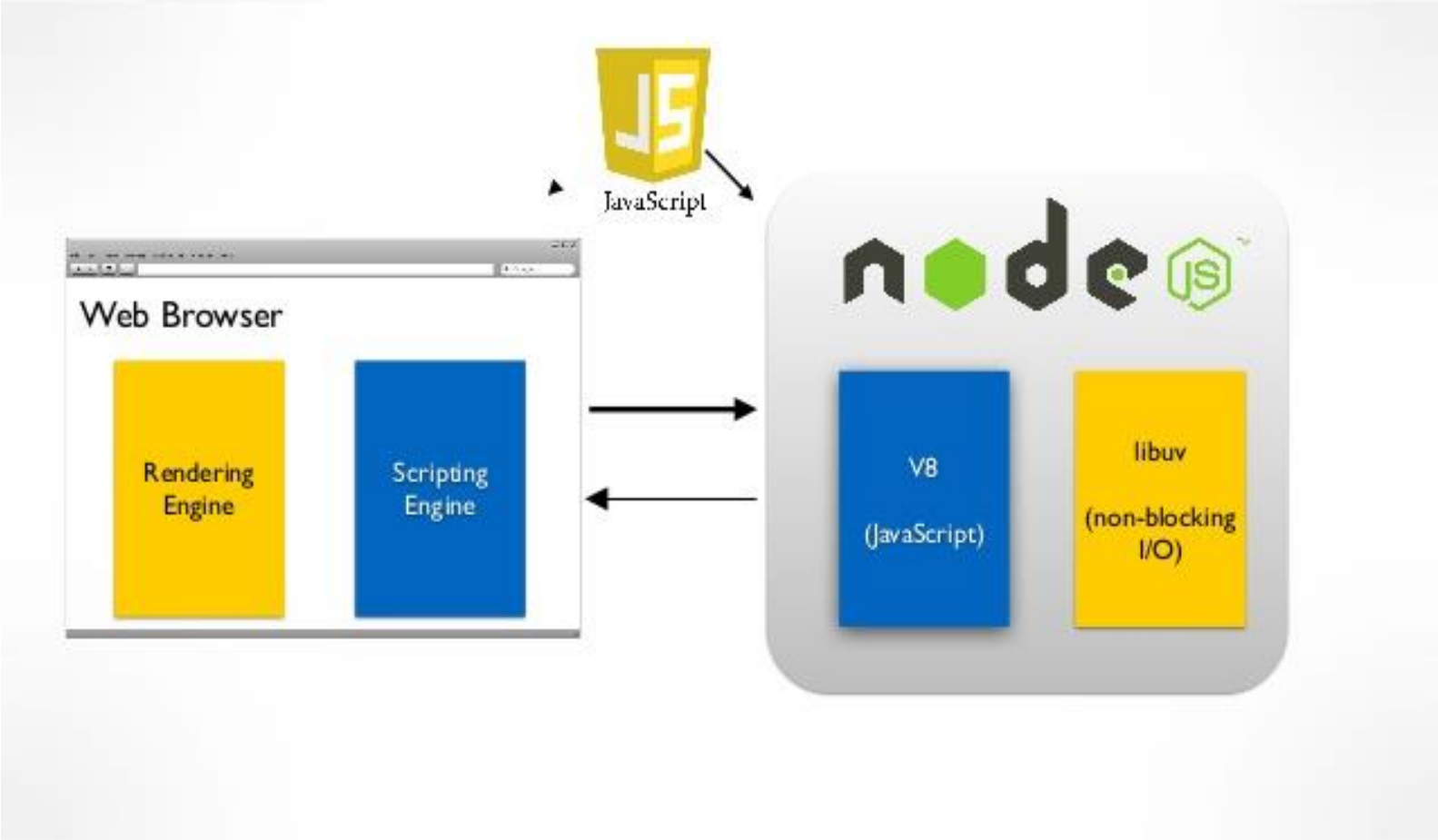
PUBLIC QUESTION #2

Why?

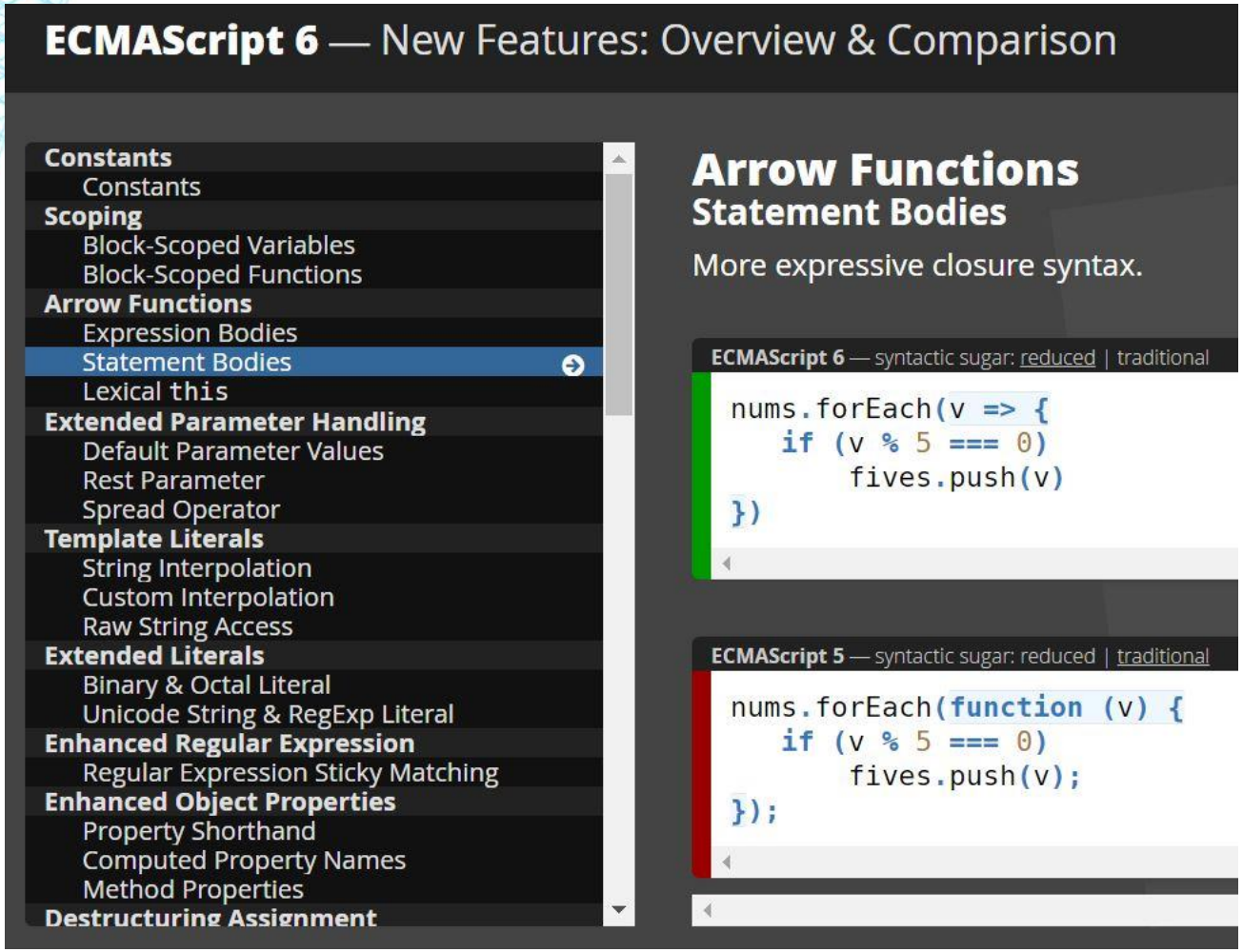
Because V8 is written in C++!

V8 – JS ENGINE DEVELOPED BY GOOGLE

- Used by Google Chrome
- Open source: https://github.com/v8/v8
- Contains bunch of C++ code
- Converts JavaScript code to machine code



ECMASCRIPT RULES EVERYTHING



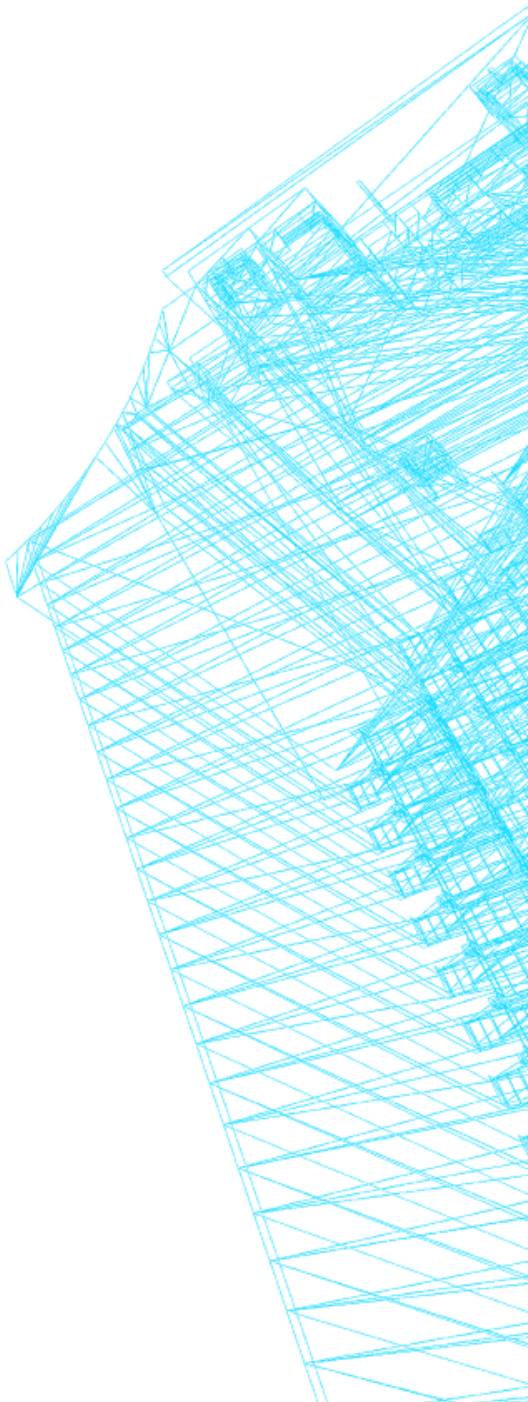
- ECMAScript: standard how Javascript should work and is translated to machine code
- JavaScript: the actual programming language
- Current version: ECMA-262, 7th Edition, ECMAScript 2016 https://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

AGENDA

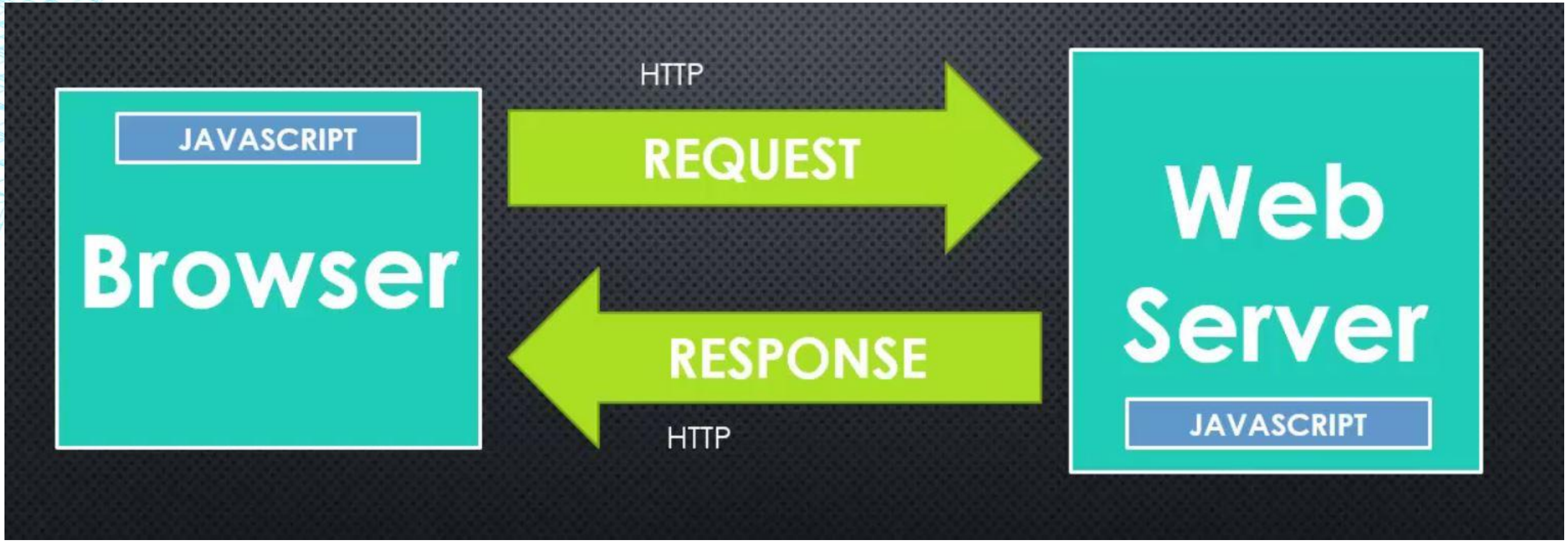
1. **Node.JS crash course – theoretical background**
    1. Node.JS and microprocessors
    2. **Node.JS basics**
    3. Parts of Node.JS
    4. Tools for Node.JS development
2. Node.JS crash course – live coding
    1. Building a complete conference website with Node.JS
    2. Scraping smashingconf.com website
3. Project roundup – scraping structured data
    1. Our Experiences, remarks, takeaways

NODE.JS

For Beginners



SERVER-CLIENT COMMUNICATION IN NODE.JS

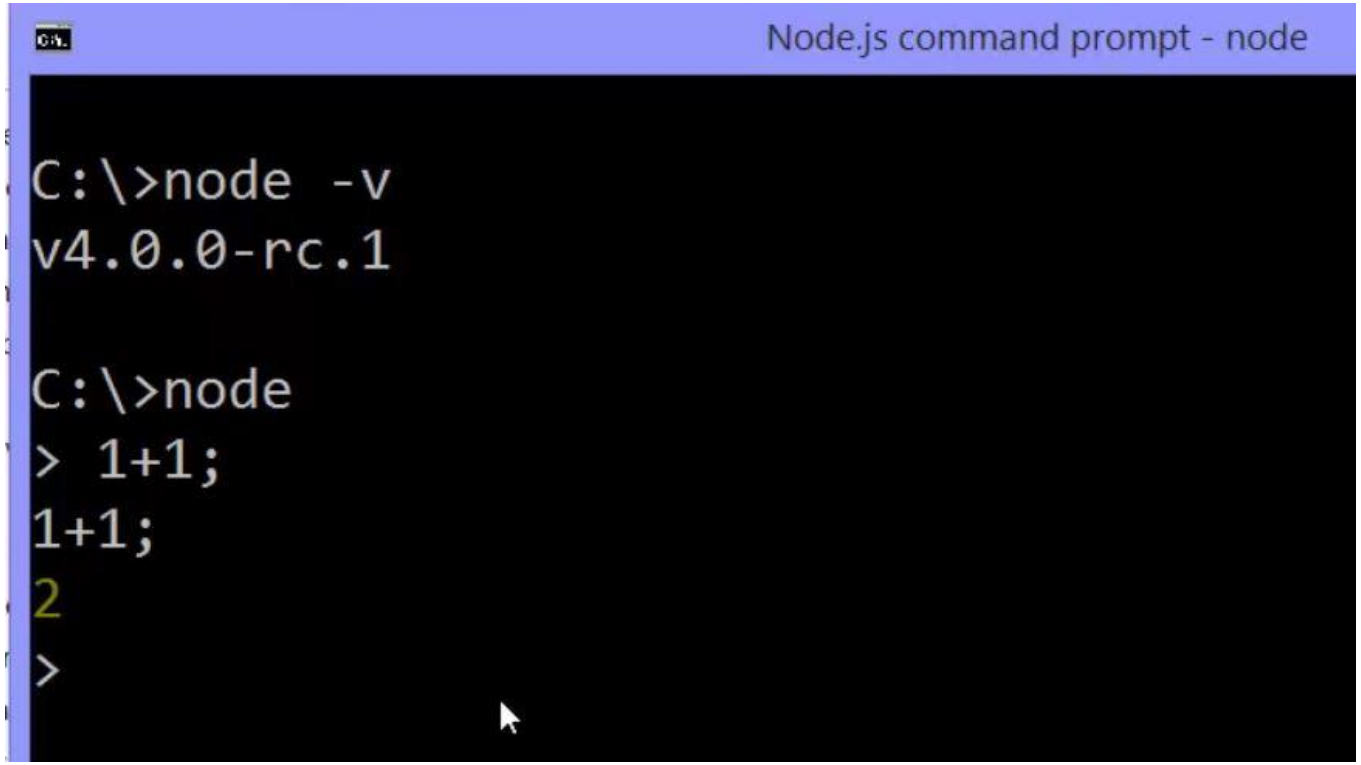


WHAT DOES JS NEED TO MANAGE A SERVER?

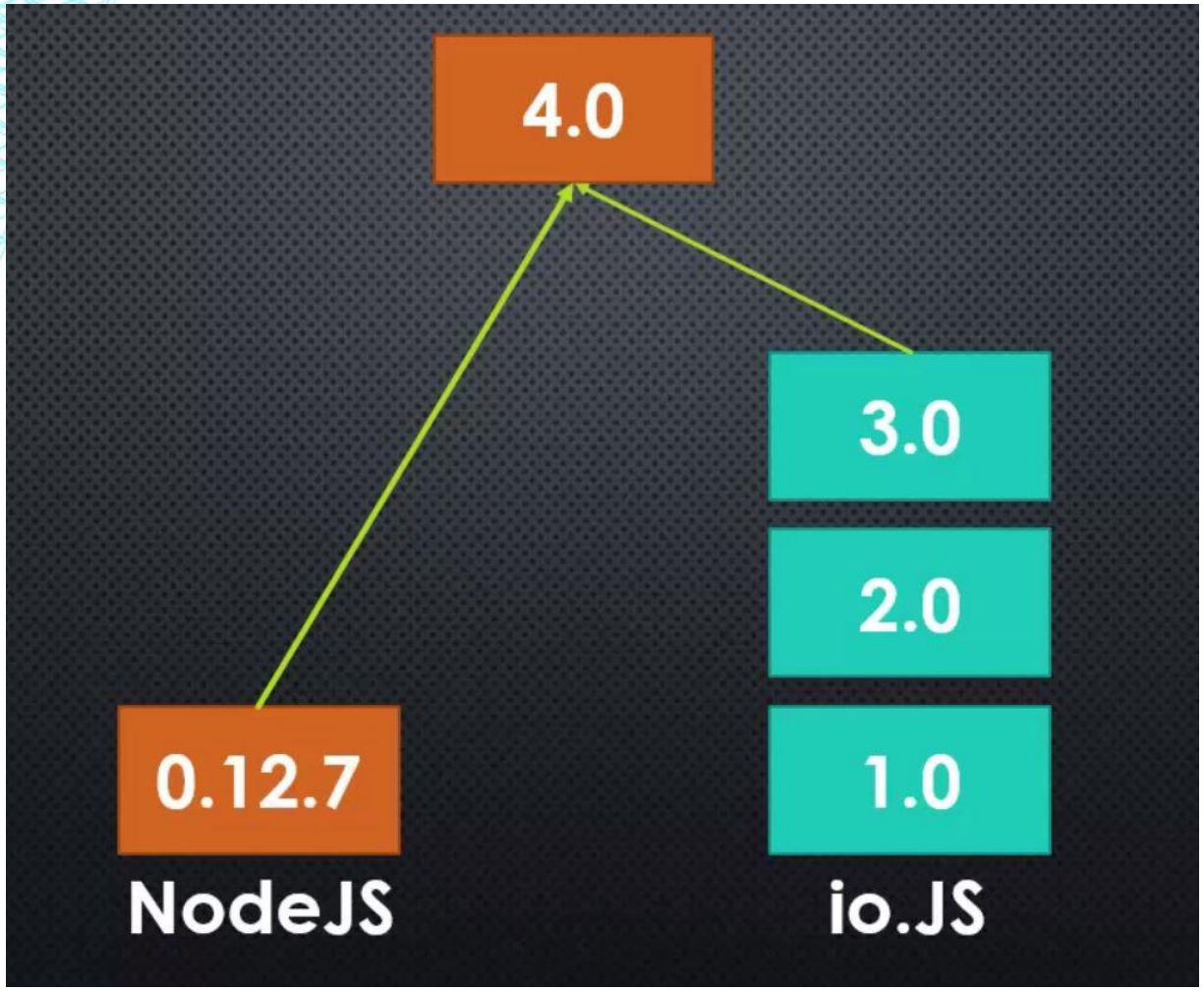
- Better ways to organize our code into reusable pieces
- Ways to deal with files
- Communicate over the internet
- Accept requests and send responses in the standard format
- Deal with databases
- Deal with work that takes a long time

YOUR FIRST NODE.JS PROGRAM

- Download Node.JS at https://nodejs.org
- Check the version with "node –v"
- The code gets compiled in the V8 engine and sent back



VERSIONS OF NODE.JS



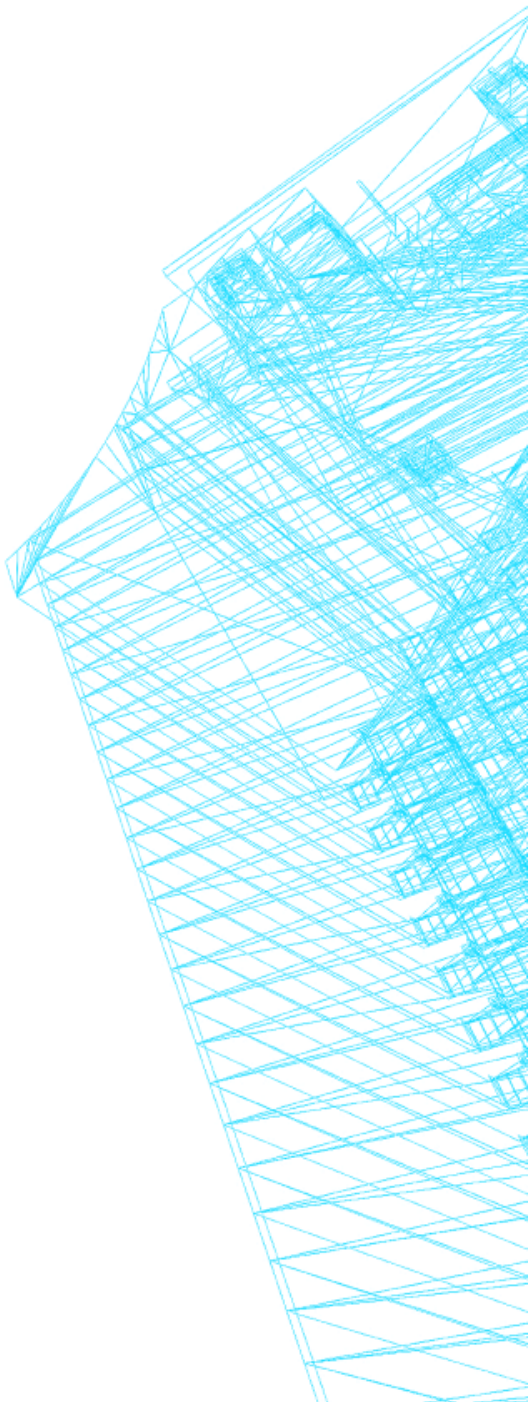
- First version of Node.JS in late 2009
- Created by the engineers at “Joyent”
- Later was renamed to io.JS, but was merged with NodeJS again 2 years ago
- Current version 7.7.2

AGENDA

1. **Node.JS crash course – theoretical background**
    1. Node.JS and microprocessors
    2. Node.JS basics
    3. **Parts of Node.JS**
    4. Tools for Node.JS development
2. Node.JS crash course – live coding
    1. Building a complete conference website with Node.JS
    2. Scraping smashingconf.com website
3. Project roundup – scraping structured data
    1. Our Experiences, remarks, takeaways

PARTS OF NODE.JS

For Intermediate



MODULES

- Reusable block of code
- Aligns with CommonJS ecosystem
- Exports specific objects, making them available for other modules

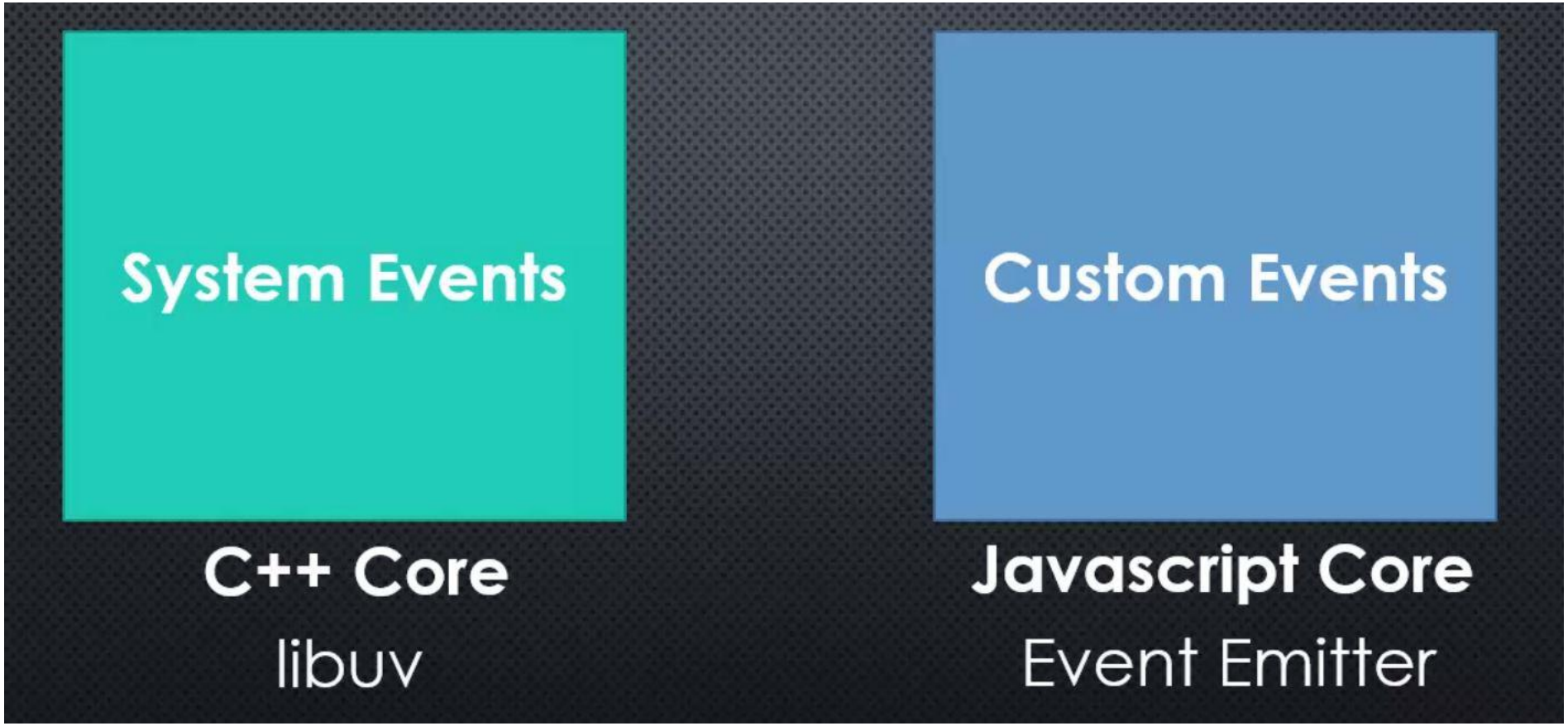
HOW MODULES ARE USED?

```
function add(){
    return x+y;
}

function subtract(){
    return x-y;
}

module.exports = {
  add: add,
  sub: subtract
};
```

This is what get exposed outside of your module's closure

```
var utils = require('./utils.js');

console.log(utils.add(1,100));
console.log(utils.sub(100,1));
```

EVENTS IN NODE.JS



EVENTS

- **Console** “Somewhere, someone said hello”, whenever the function **greet()** is called in our code

• app.js - nodejs - Visual Studio Code

• app.js

```
1 var Emitter = require('./emitter');
2
3 var emtr = new Emitter();
4
5 emtr.on('greet', function() {
6     console.log('Somewhere, someone said hello.');
7 });
```

PUBLIC QUESTION #3

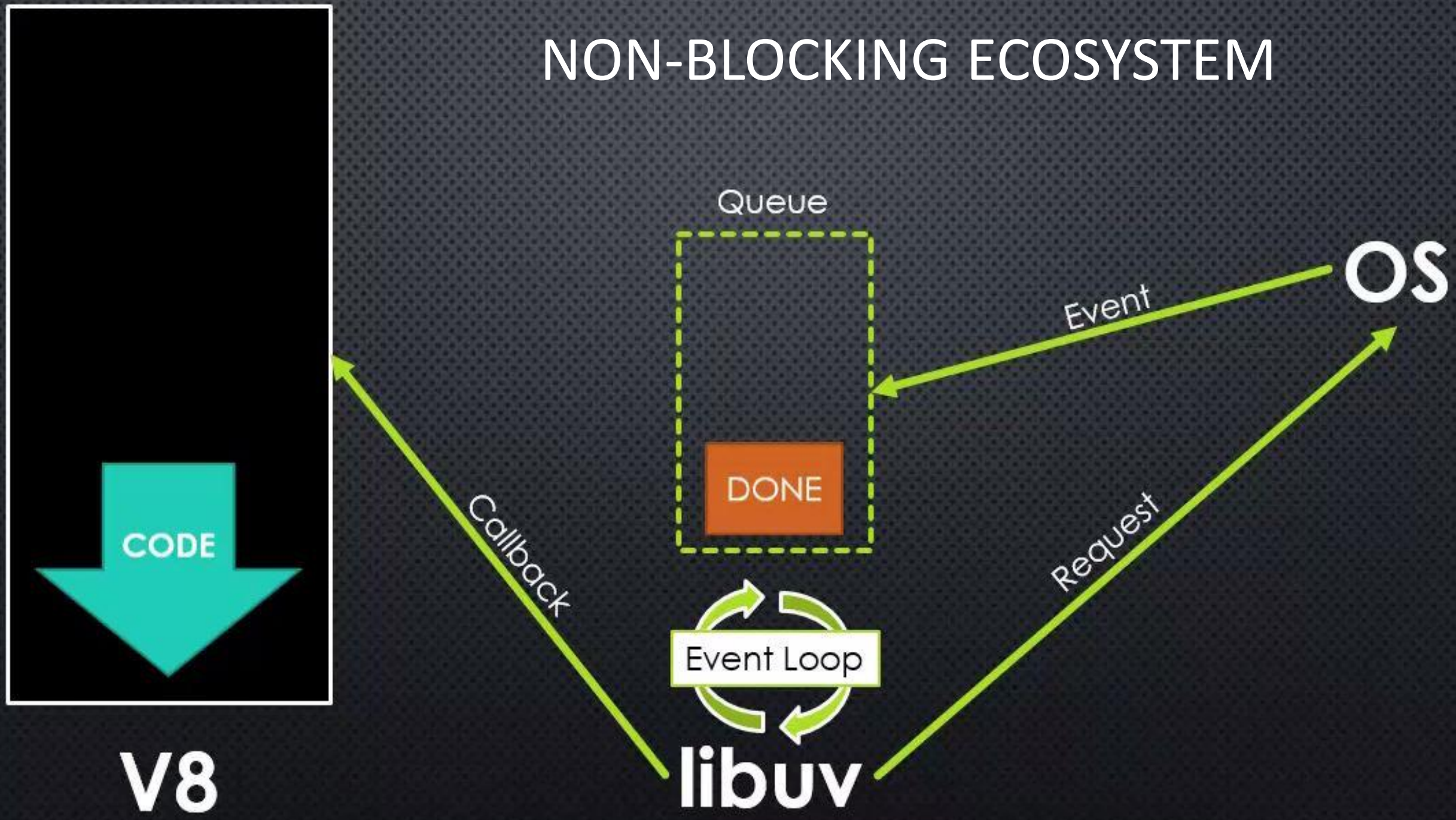
Is JavaScript synchronous or asynchronous?

Synchronous

SYNCHRONOUS CODE VS ASYNCHRONOUS

- Asynchronous process or asynchronous program is when more than one process is running simultaneously
- In JS, only one line of code executed at a time
- However, NodeJS is asynchronous

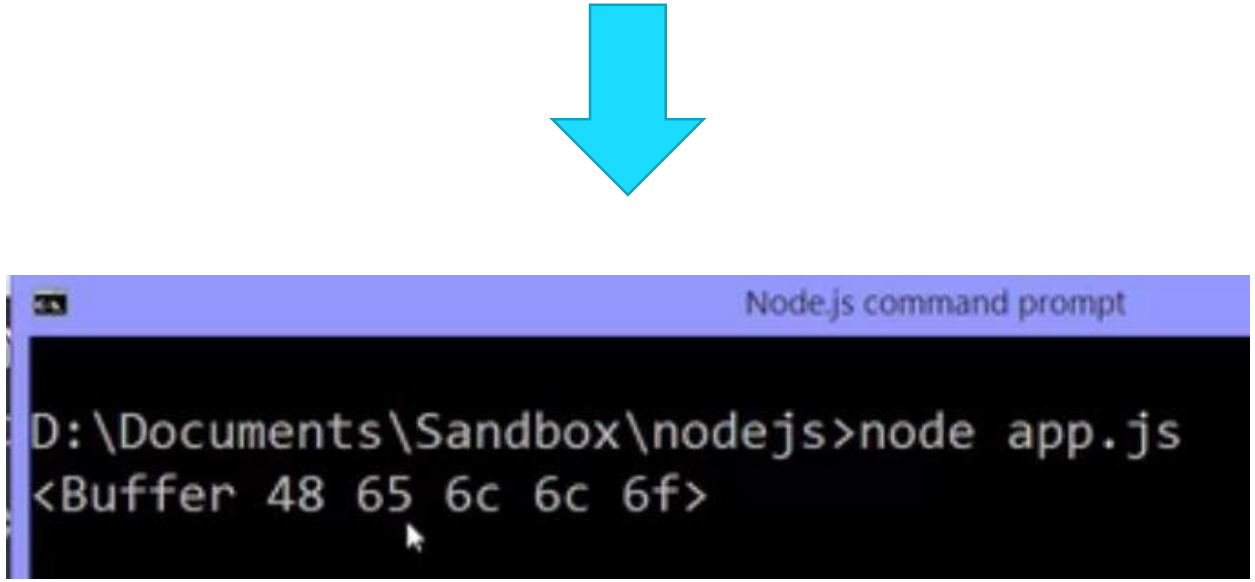
NON-BLOCKING ECOSYSTEM



BUFFERS AND FS

- Reads and manipulates streams of binary data
- Work in background of the “fs” library
- We can also write data into buffers by ourselves!

```
var buf = new Buffer('Hello', 'utf8');
console.log(buf);
```



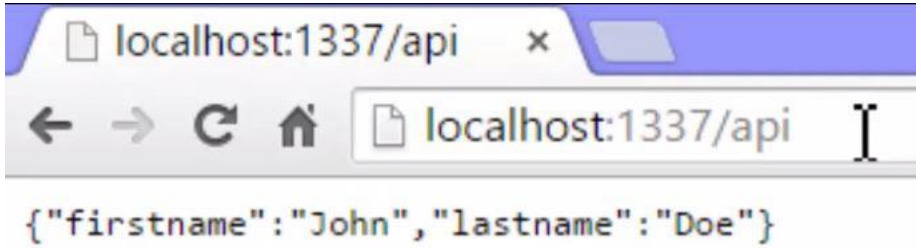
CALLBACKS

- A Callback is an inherent part of Node.JS
- Called whenever the main function terminates
- Returns error as its first parameter and response as a second parameter

HTTP REQUESTS AND RESPONSES

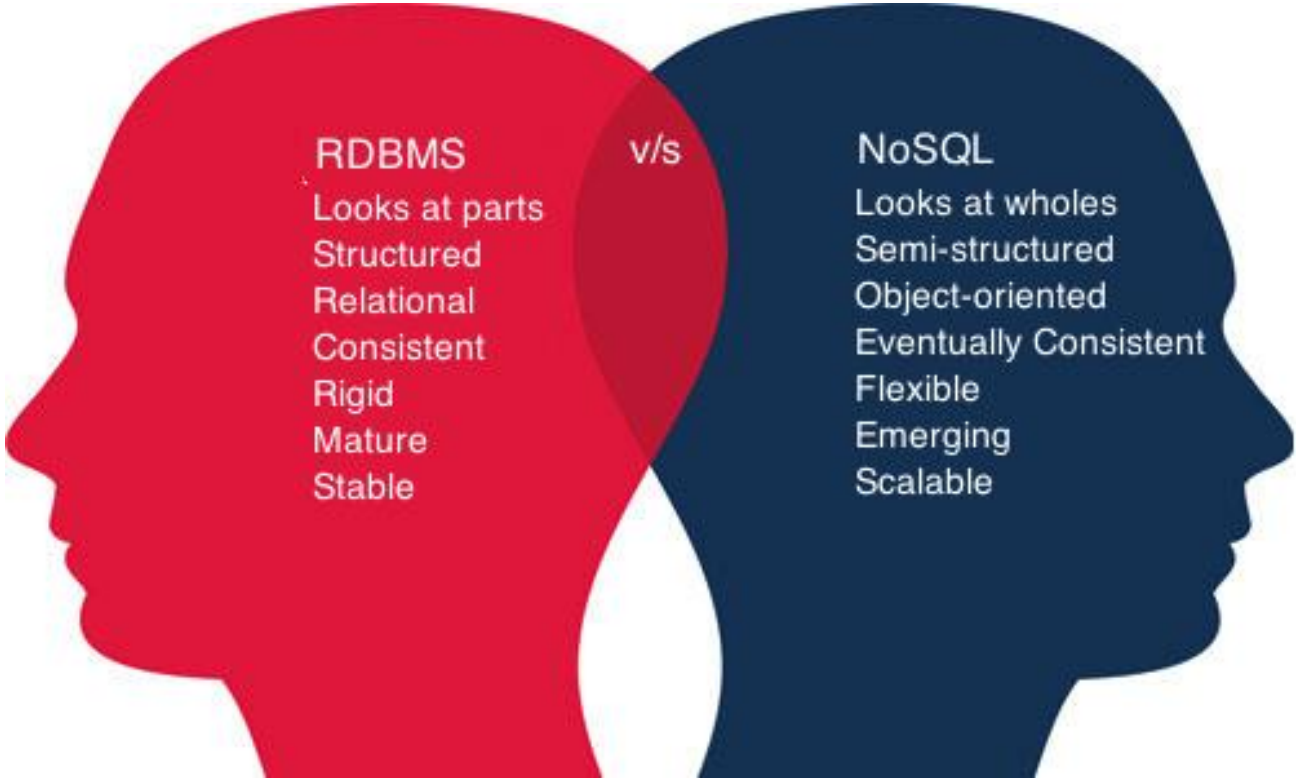
app.js

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

http.createServer(function(req, res) {

    if (req.url === '/api') {
        res.writeHead(200, { 'Content-Type': 'application/json' });
        var obj = {
            firstname: 'John',
            lastname: 'Doe'
        };
        res.end(JSON.stringify(obj));
    }

}).listen(1337, '127.0.0.1');
```



NODE.JS AND DATABASES

- SQL and NoSQL possible
- Integrates easily with MongoDB that forms a part of the MEAN stack
- NoSQL – provides flexibility, ideal to hold the web data e.g. : photos , videos, blogs
- Mongoose – simple solution to model application data



WHAT DOES JS NEED TO MANAGE A SERVER? (AGAIN)

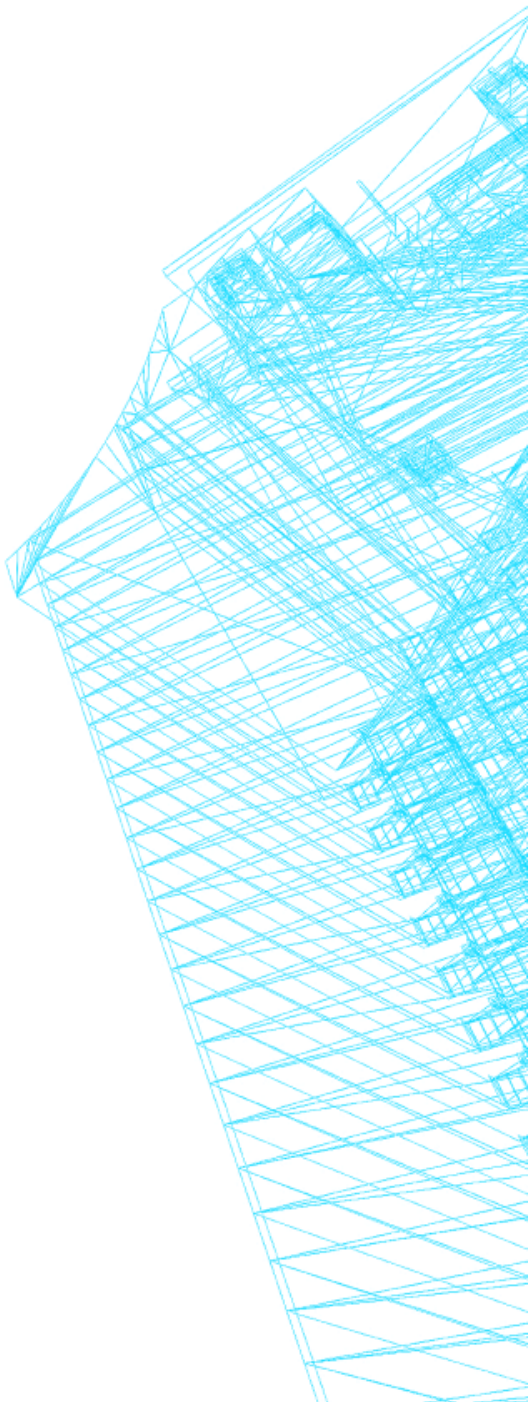
- Better ways to organize our code into reusable pieces ✓
- Ways to deal with files ✓
- A way to deal with work that takes a long time ✓
- The ability to communicate over the internet ✓
- The ability to accept requests and send responses in the standard format ✓
- Ways to deal with databases ✓

AGENDA

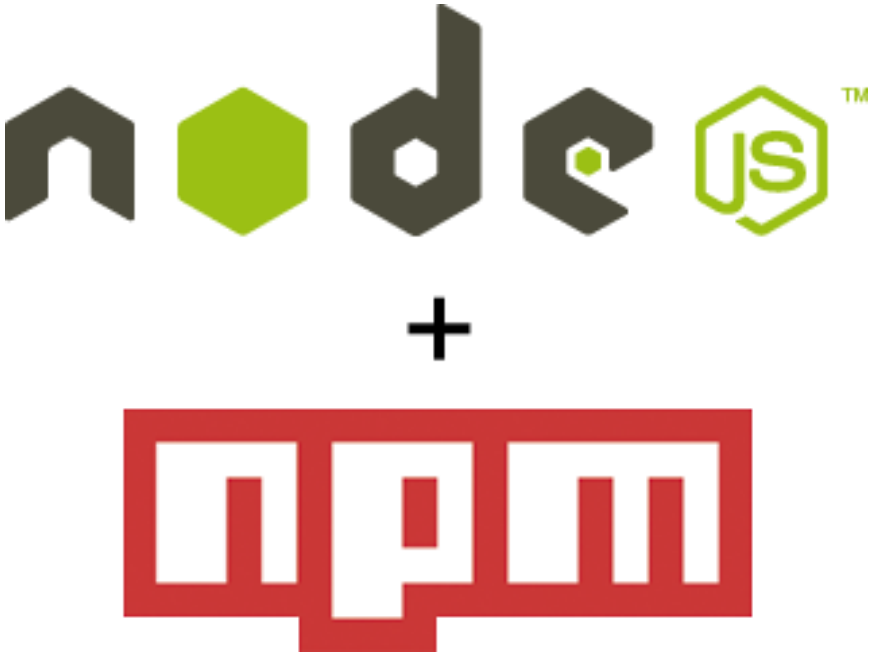
1. **Node.JS crash course – theoretical background**
    1. Node.JS and microprocessors
    2. Node.JS basics
    3. Parts of Node.JS
    4. **Tools for Node.JS development**
2. Node.JS crash course – live coding
    1. Building a complete conference website with Node.JS
    2. Scraping smashingconf.com website
3. Project roundup – scraping structured data
    1. Our Experiences, remarks, takeaways

PACKAGE MANAGERS, TASK RUNNERS, FRAMEWORKS…

For Advanced



NPM – NODE PACKAGE MANAGER



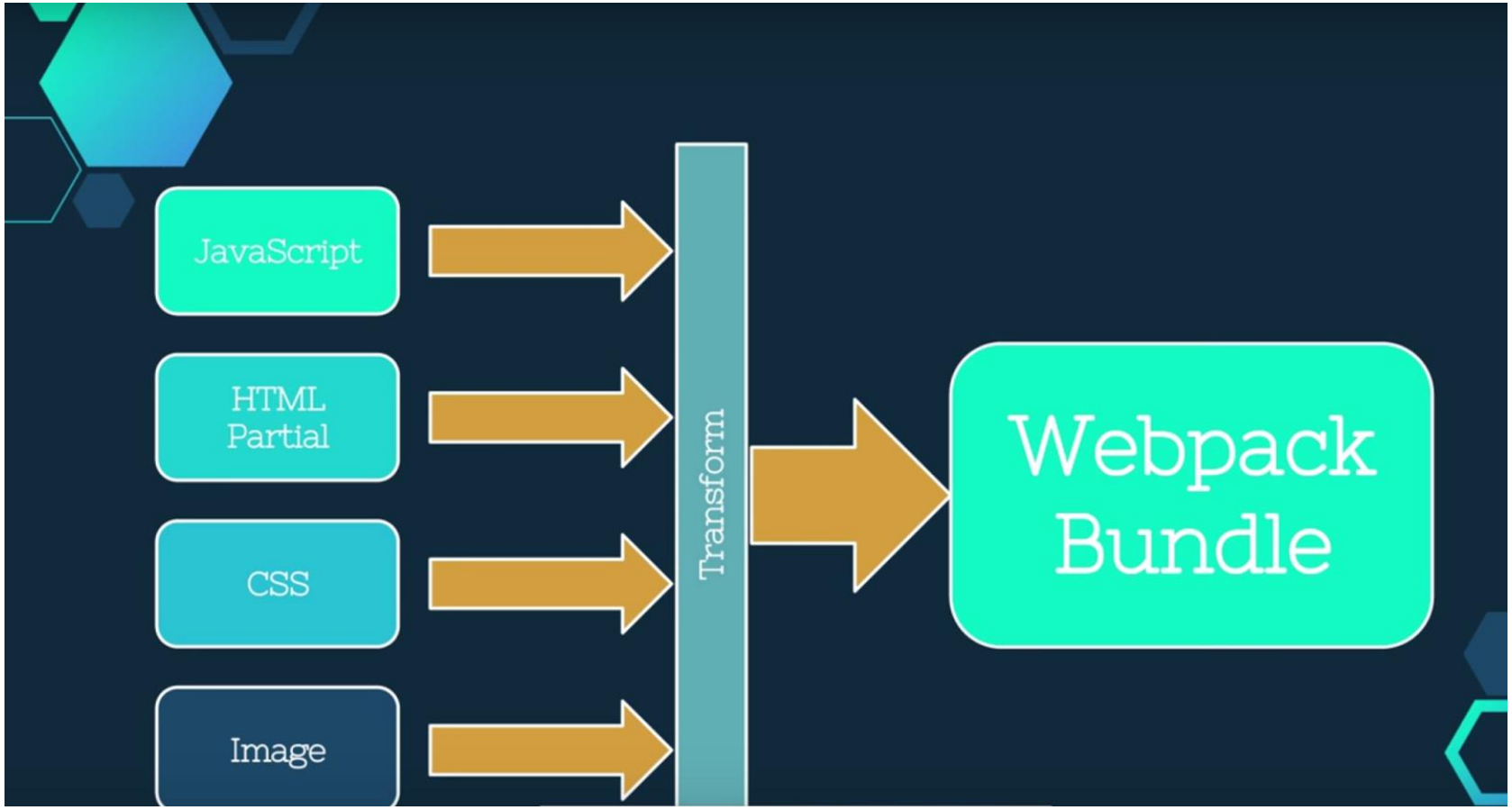
- NPM manages dependencies
- Updates packages when needed
- npm makes it easy for JavaScript developers to share and reuse code

HOW VERSIONING WORKS

1. Update **from 1.7.0 to 1.7.1** - some bugs were fixed, your code will work fine
2. Update **from 1.7.1 to 1.8.0** - some new features were added, your code will still work just fine
3. Update **from 1.8.0 to 2.0.0** - big changes, your code will probably break

WEBPACK

- Converts your static assets to a bundle
- Supports hot module reloading
- Works very well with React framework



GULP

```
[14:01:18] Using gulpfile C:\[redacted]\formdata\gulpfile.js
[14:01:18] Starting 'watch-test'...
[14:01:18] Finished 'watch-test' after 45 ms
[14:01:32] Starting 'test'...


  Home page
    - should load the page properly
    - should navigate to login
    - should navigate to sign up
    - should load analytics


  0 passing (13ms)
  4 pending

[14:01:32] Finished 'test' after 70 ms
```

- Automates the common tasks while building applications
- With Gulp you can: minimize CSS, convert LESS to CSS, make certain modifications on certain conditions

PACKAGES WE USE

- ExpressJS
- Nodemon
- Moment
- Request

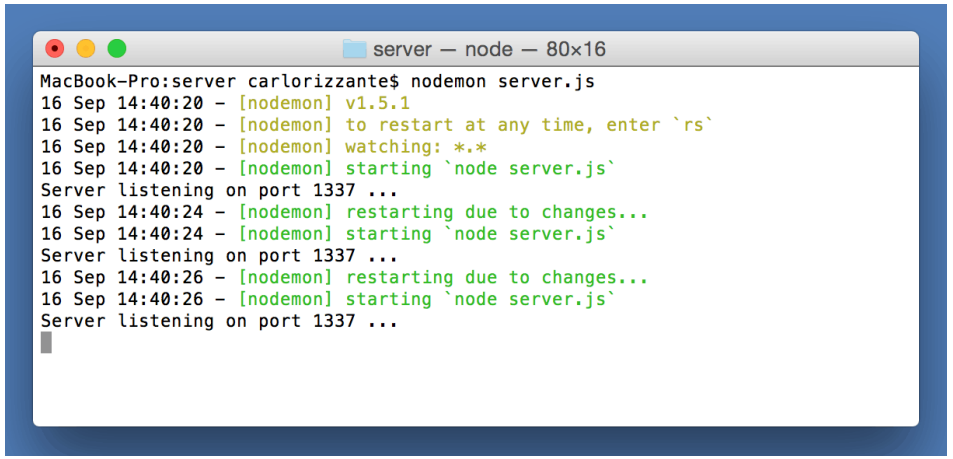
EXPRESS

- Helps to organize your app into MVC structure
- Allows usage of any templating language
- Part of the MEAN stack
- Routing made easyyyy…

```
// GET method route
app.get('/', function (req, res) {
  res.send('GET request to the homepage')
})

// POST method route
app.post('/', function (req, res) {
  res.send('POST request to the homepage')
})
```

NODEMON

- Watches for changes in the file directory and restarts the app if needed



MOMENT

Format Dates

```
moment().format('MMMM Do YYYY, h:mm:ss a'); // March 10th 2017, 4:49:48 pm
moment().format('dddd');                    // Friday
moment().format("MMM Do YY");               // Mar 10th 17
moment().format('YYYY [escaped] YYYY');     // 2017 escaped 2017
moment().format();                          // 2017-03-10T16:49:48+01:00
```

- Parses, manipulates, validates and formats dates

REQUIRE

```
var request = require('request');
request('http://www.google.com', function (error, response, body) {
  console.log('error:', error); // Print the error if one occurred
  console.log('statusCode:', response && response.statusCode); // Print the response
status code if a response was received
  console.log('body:', body); // Print the HTML for the Google homepage.
});
```

- Library that allows us to make simple HTTP calls
- It returns the DOM structure of the website that could be used e.g. for scraping
- Important part of our work

AGENDA

1. Node.JS crash course – theoretical background
    1. Node.JS and microprocessors
    2. Node.JS basics
    3. Parts of Node.JS
    4. Tools for Node.JS development
2. **Node.JS crash course – live coding**
    1. Building a complete conference website with Node.JS
    2. Scraping smashingconf.com website
3. Project roundup – scraping structured data
    1. Our experiences, remarks, takeaways

AGENDA

1. Node.JS crash course – theoretical background
    1. Node.JS and microprocessors
    2. Node.JS basics
    3. Parts of Node.JS
    4. Tools for Node.JS development
2. Node.JS crash course – live coding
    1. Building a complete conference website with Node.JS
    2. Scraping smashingconf.com website
3. **Project roundup – scraping structured data**
    1. Our experiences, remarks, takeaways

THE PROJECT SCOPE

- At first there was…

?

THE PROJECT SCOPE

- Extract data from structured online resources:
  - DBPedia
  - WorldCat
  - MusicBrainz
  - IMSLP
  - …
- Define a vocabulary for Team 2
- Set up a database
- Populate the database with the structured data
- Create a REST API to access the database

PROBLEMS ALONG THE WAY

Extract data from structured online resources

- Some databases are harder to scrape than others
- Obviously, every script differs since the structure of each website differs
- And the winner in the category “most effort” is…

PROBLEMS ALONG THE WAY

PROBLEMS ALONG THE WAY

Issues with DBPedia:

- Tried Sparkle queries first, failed
- No structured overview, just some lists (17th_century_classical_musicians, classical_musicians_by_nationality, classical_musicians_by_instruments_and_nationality, …)
- All lists differ in structure
- Even the resulting artist pages differ in structure (i.e. 4 different fields, that **could** contain “date of birth”)
- Some fields missing, i.e. nationality which could sometimes be extracted from the list URL one level above (if list was named “American_classical_pianists”)

PROBLEMS ALONG THE WAY

- Define a vocabulary for Team 2:
  - What is meant by the term “vocabulary”
    - Words that are in context with “classical music”?
    - All the data we scraped including all attributes?
    - A plain list of entity names regardless of their meaning?
- Set up a database:
  - Which database is best for our purposes ?
  - How do we set up our PostgreSQL db?

PROBLEMS ALONG THE WAY

- Populate the database with the structured data:
  - handle differences in structure since data is aggregated from different databases
  - …and then do it all again with Sequelize
- Create a REST API to access the database:
  - Learn about REST APIs
  - Write the code
  - … and then do it all again with Sequelize + Swagger

LESSONS LEARNED

- It is tough to work towards a goal if you do not know exactly where you are heading
- Many helpful packages/frameworks, but you need to find and learn them first
- Tools should be known upfront, otherwise a lot of additional effort
- Earlier technical meeting to clarify questions, such that everyone has the same picture in mind

THANKS FOR YOUR ATTENTION!

WE WOULD BE HAPPY TO ANSWER SOME QUESTIONS

JAVASCRIPT ON THE SERVER. A NODE.JS CRASH COURSE

Group A

Lukas Navickas, Shilpa Ghanashyam Gore, Angelin Rashmi, Tim Henkelmann