
2016
THAT
CONFERENCE
SUMMER CAMP FOR GEEKS
TM

THANK YOU, THAT CONFERENCE SPONSORS!

Microsoft

Quicken Loans
Engineered to Amaze

IBM

inrule TECHNOLOGY

COLLABORATIVE CDSC

SKYLINE
see beyond your it

omni resources

TARGET

JET BRAINS

jibo

Northwestern Mutual

twilio

Node.js Crash Course

```
let trustMe = {
    has: ["Beard", "Motorcycle"],
    consumes: ["Bacon", "Caffeine"]
};
```

David Neal @ReverentGeek

JavaScript...

...has won the Web.

Scott Hanselman

Atwood's Law:

Any application that can be written in JavaScript...

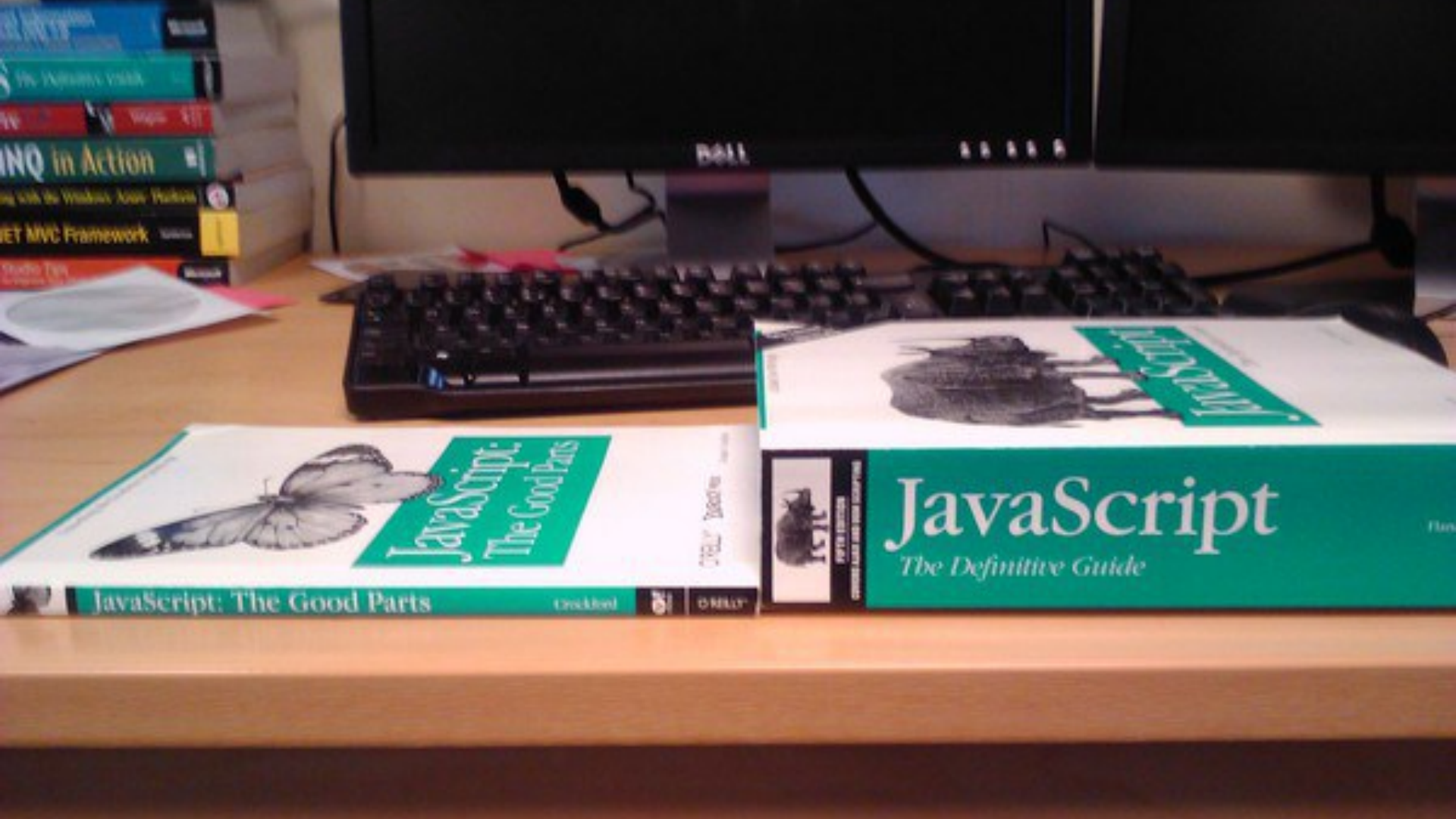
...will eventually be written in JavaScript

(search for "jslinux")

JavaScript...

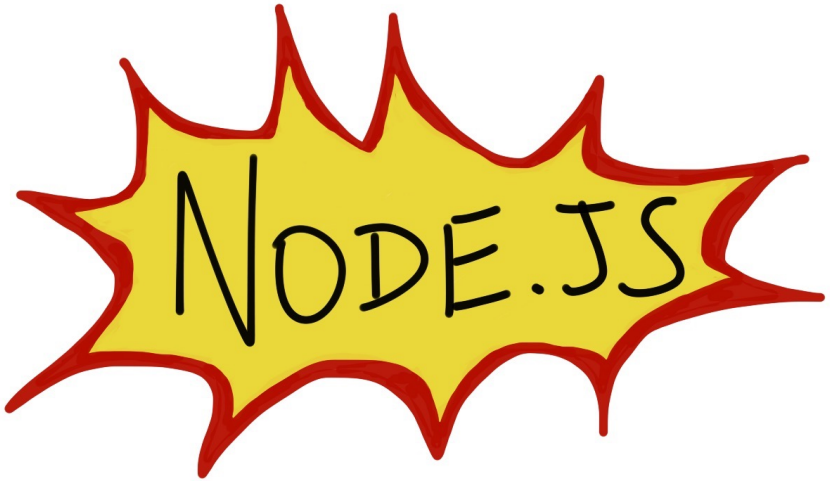
...is tragically important.

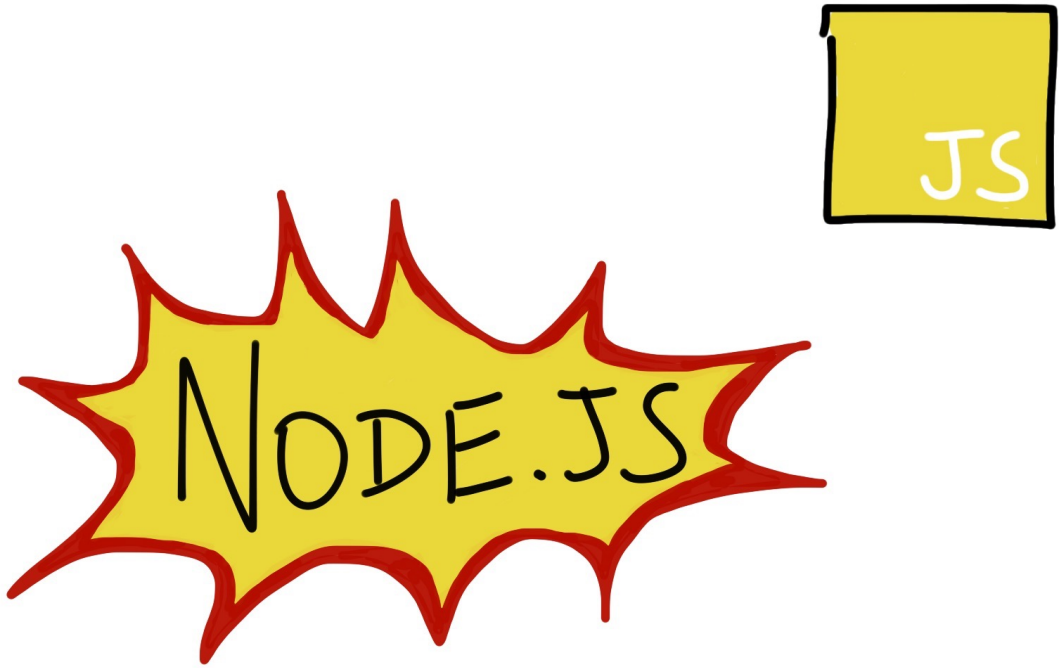
Douglas Crockford

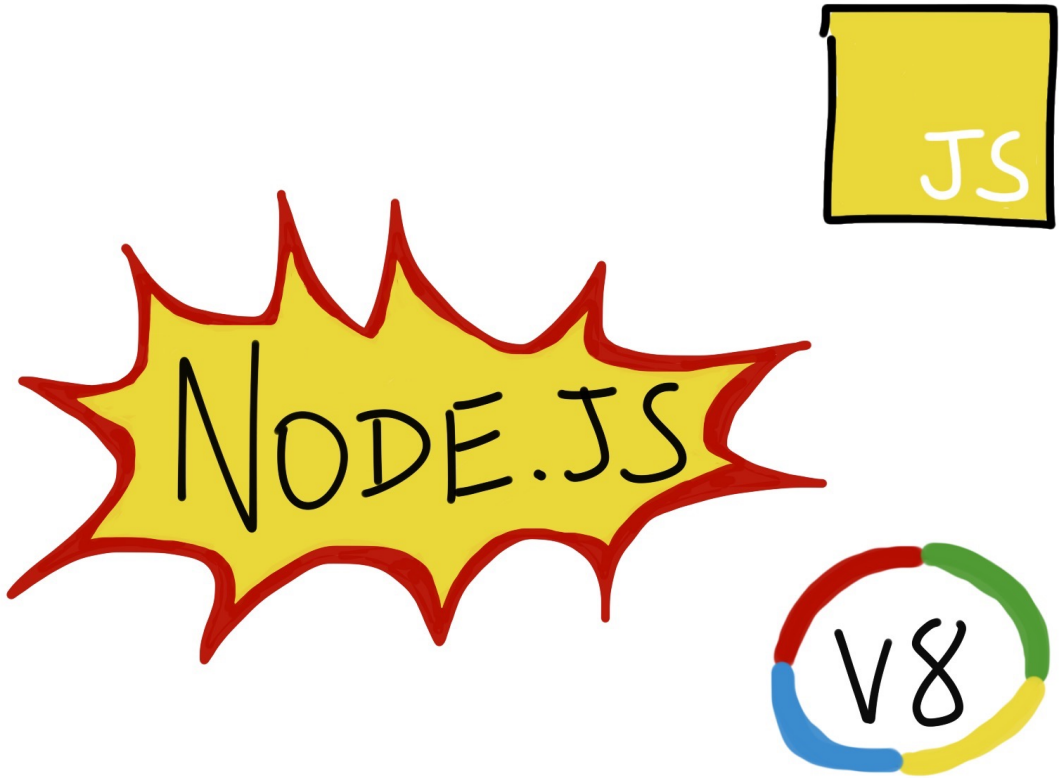

JavaScript: The Good Parts
JavaScript
The Definitive Guide

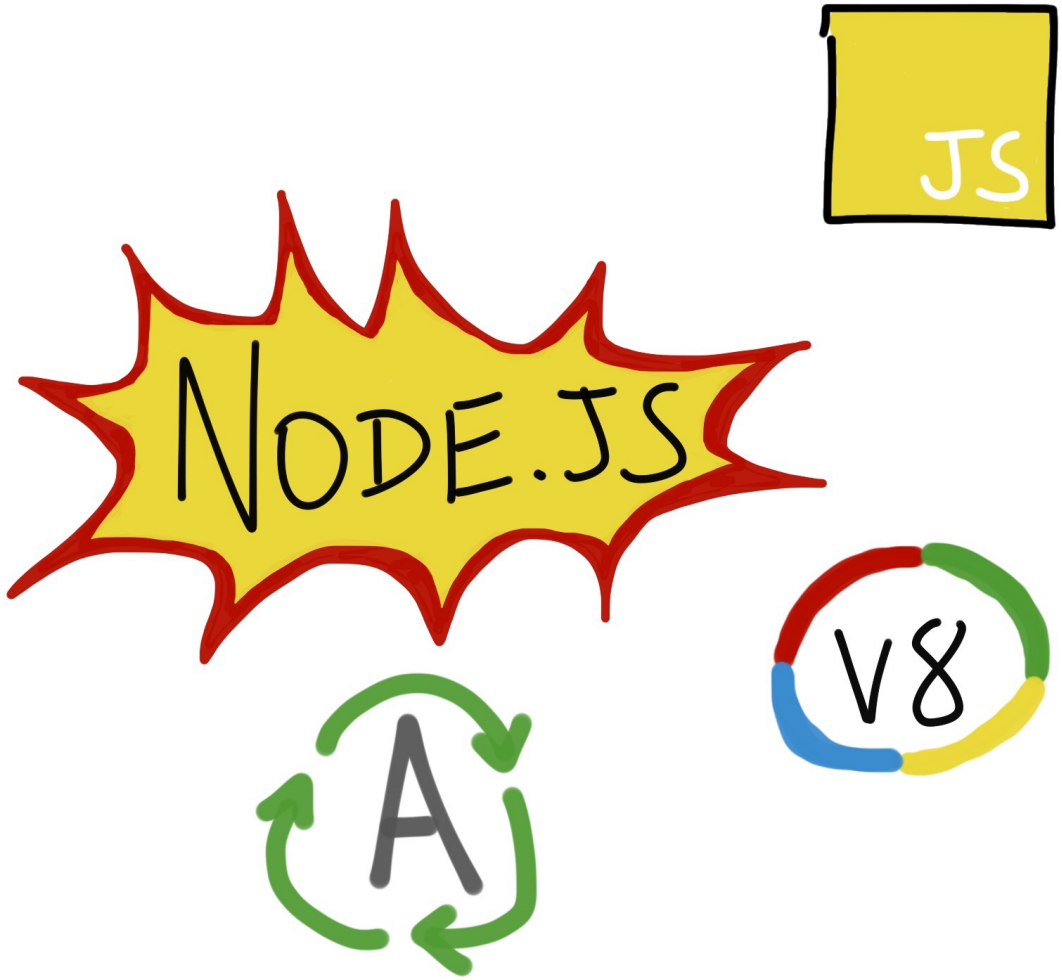
Up Ahead

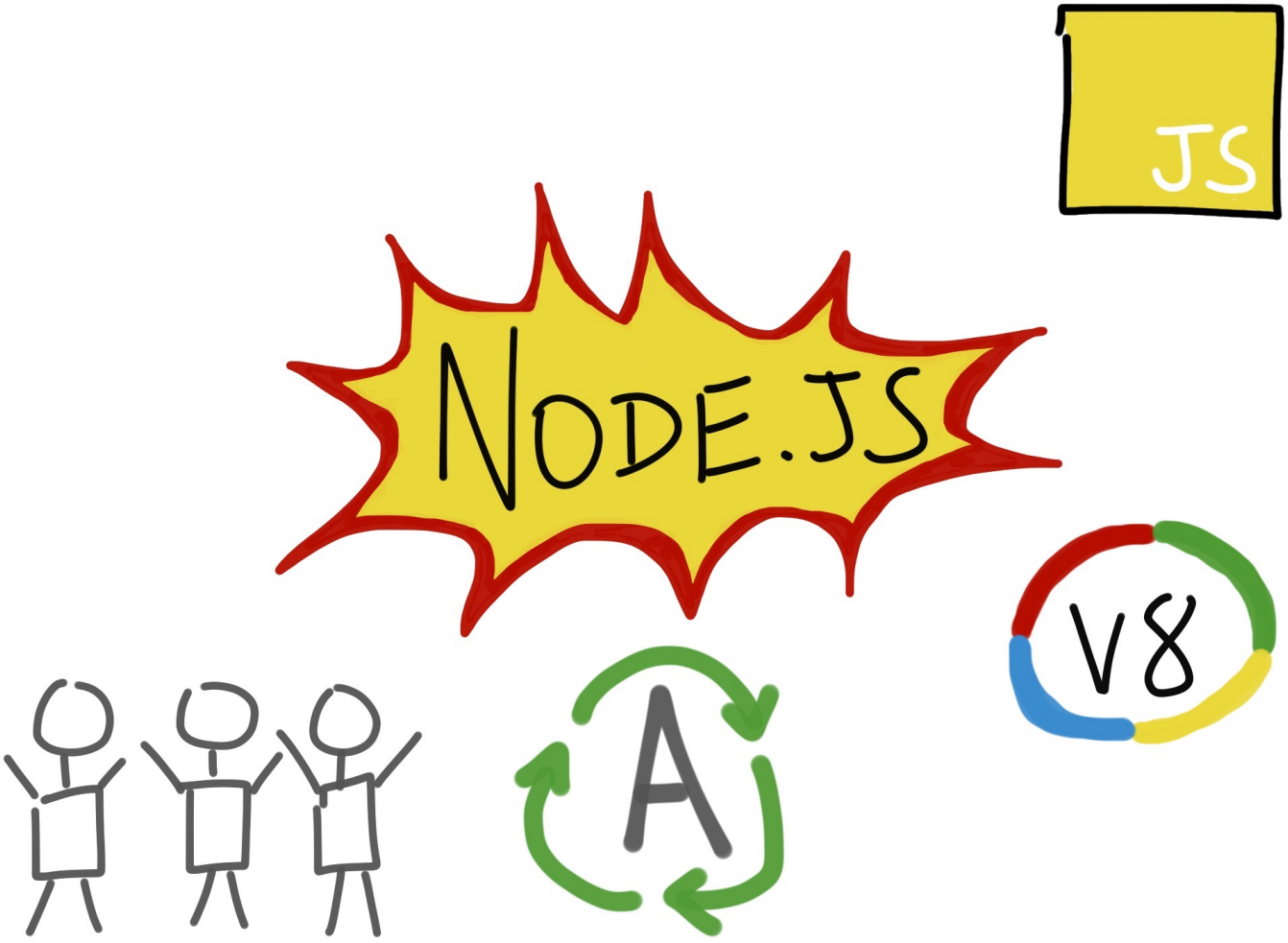
- Why Node.js?
- Crash course
- Tools and frameworks
- Integration strategies

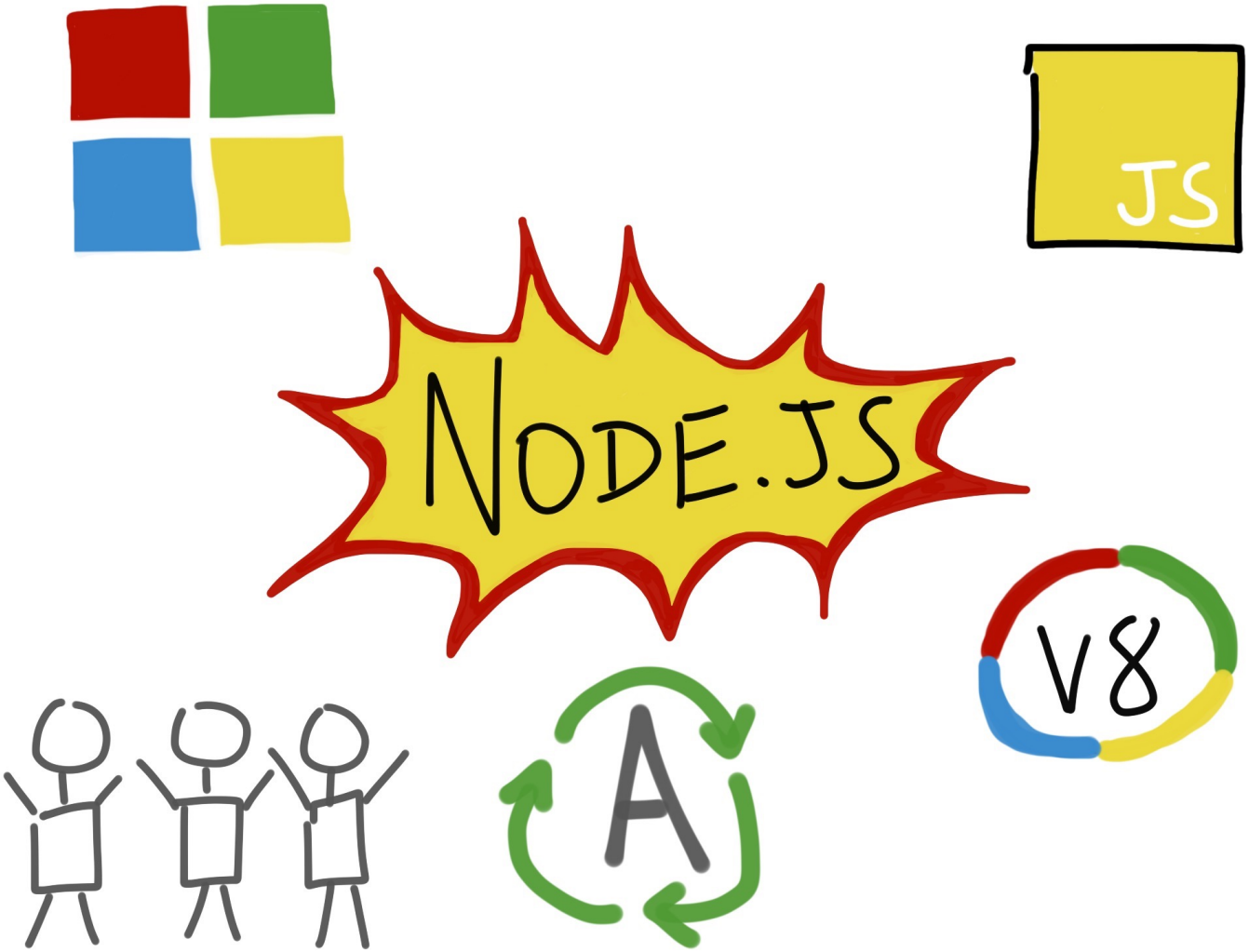

NODE.JS


JS
NODE.JS


JS
NODE.JS
V8

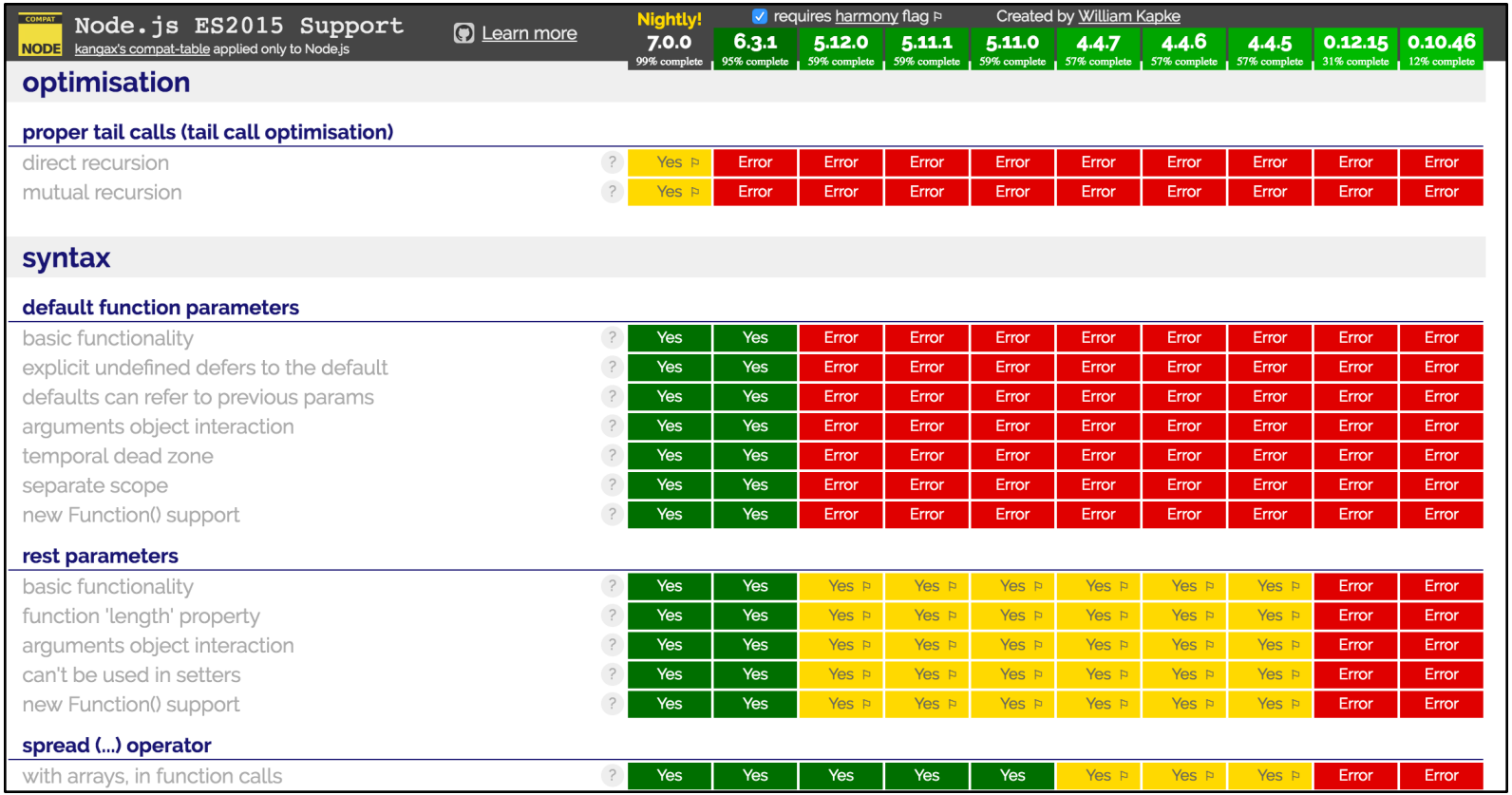

JS
NODE.JS
V8
A


JS
NODE.JS
V8
A


JS
NODE.JS
V8
A

Not Your Dad's JavaScript

ES 2016 (ES6) Support
http://node.green

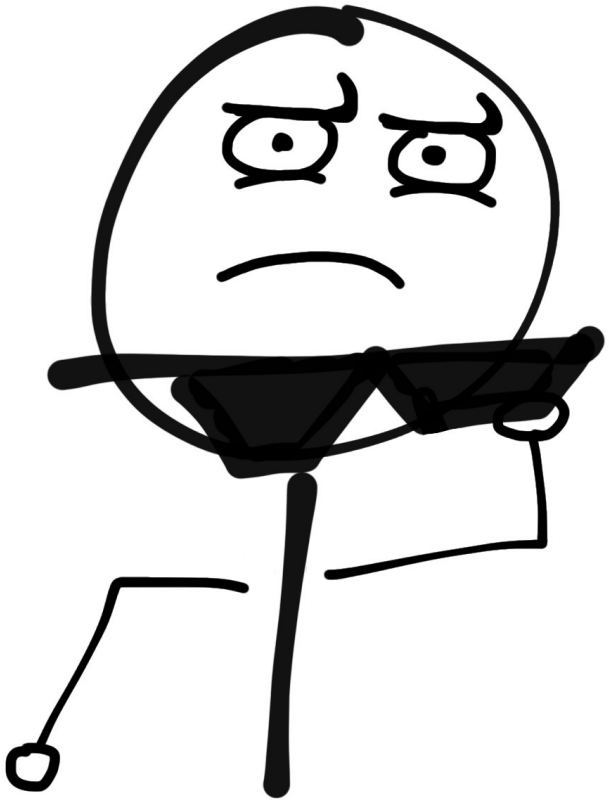


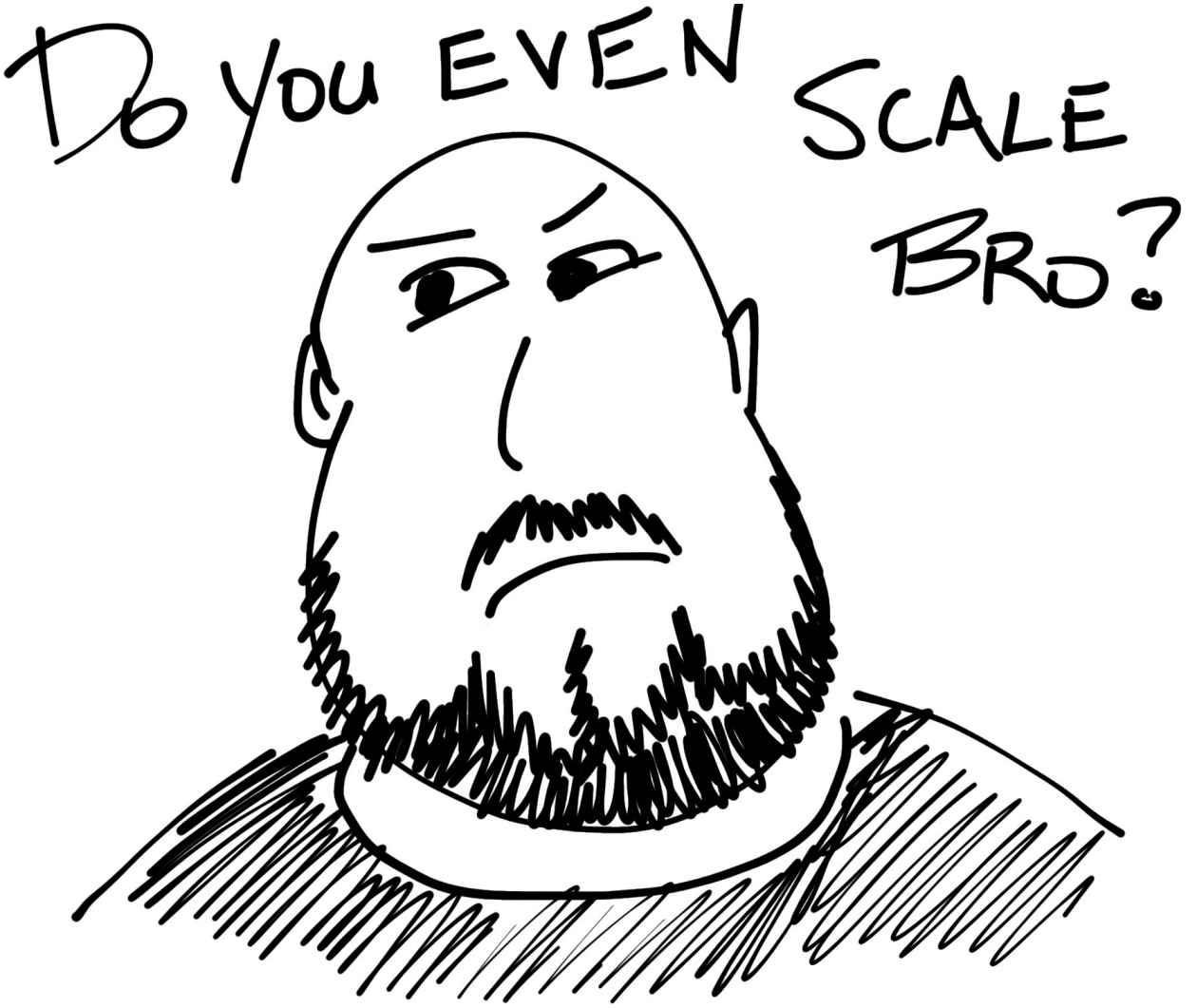
COMPAT NODE — Node.js ES2015 Support
kangax's compat-table applied only to Node.js
Learn more
Nightly! 7.0.0 — 99% complete
requires harmony flag
Created by William Kapke

	Nightly! 7.0.0 (99% complete)	6.3.1 (95% complete)	5.12.0 (59% complete)	5.11.1 (59% complete)	5.11.0 (59% complete)	4.4.7 (57% complete)	4.4.6 (57% complete)	4.4.5 (57% complete)	0.12.15 (31% complete)	0.10.46 (12% complete)
optimisation										
proper tail calls (tail call optimisation)										
direct recursion	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
mutual recursion	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
syntax										
default function parameters										
basic functionality	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error
explicit undefined defers to the default	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error
defaults can refer to previous params	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error
arguments object interaction	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error
temporal dead zone	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error
separate scope	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error
new Function() support	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error
rest parameters										
basic functionality	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error
function 'length' property	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error
arguments object interaction	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error
can't be used in setters	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error
new Function() support	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error
spread (...) operator										
with arrays, in function calls	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error

What's our story?

- ASP.NET MVC, C#
- SQL Server + NHibernate
- JavaScript + JQuery

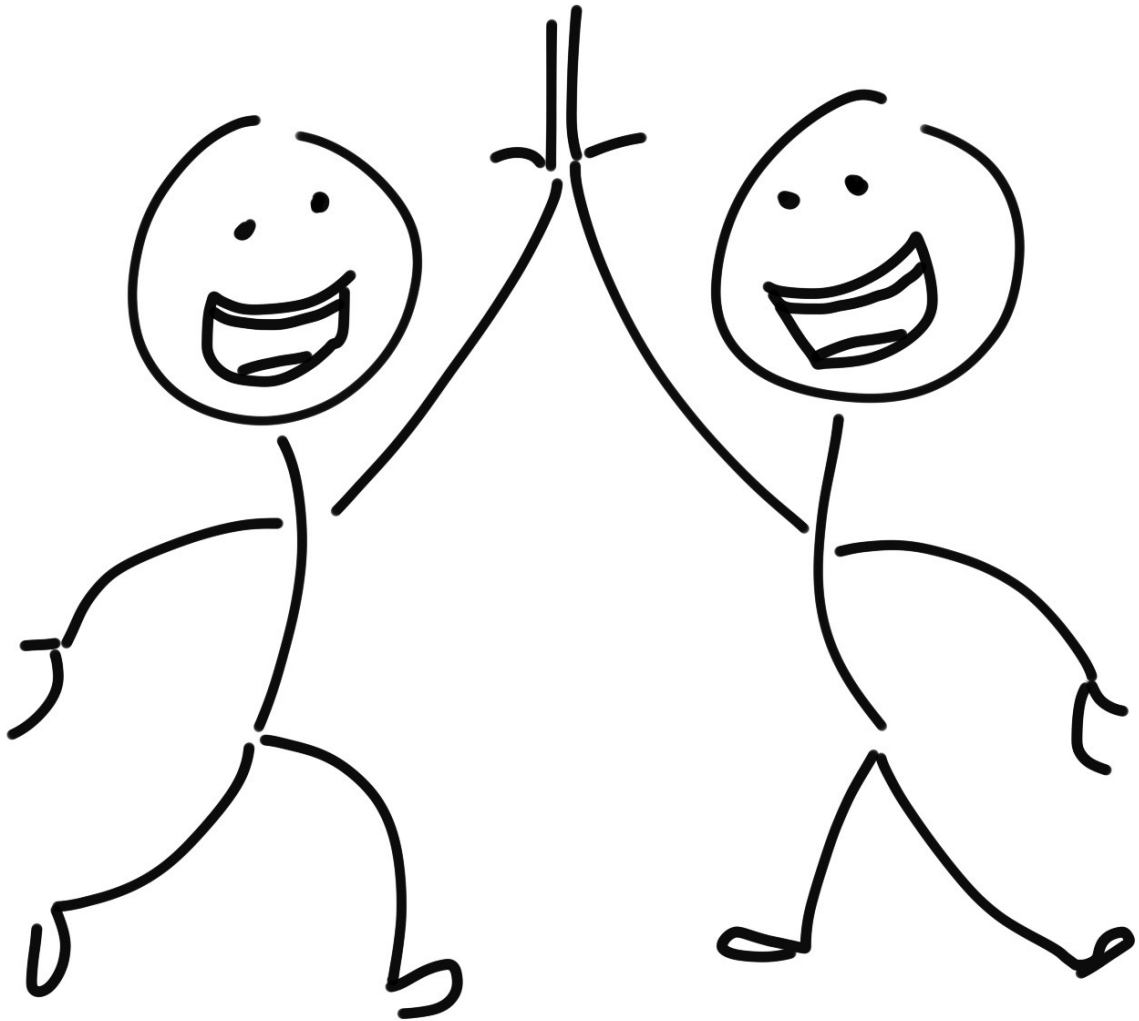



Do you even scale bro?

What's our story?

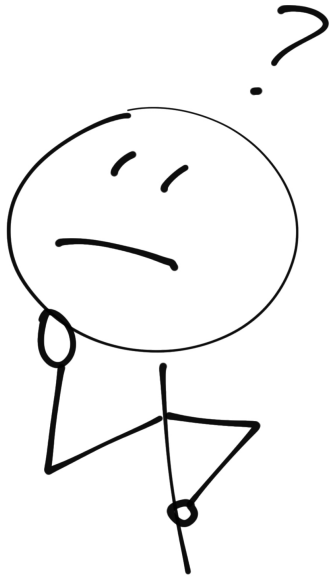
- JavaScript
- Recruiting
- Productive, less friction
  - Testing
  - Microservices
  - RabbitMQ, riak, redis
  - Cross-platform

leankit



Why Node.js?

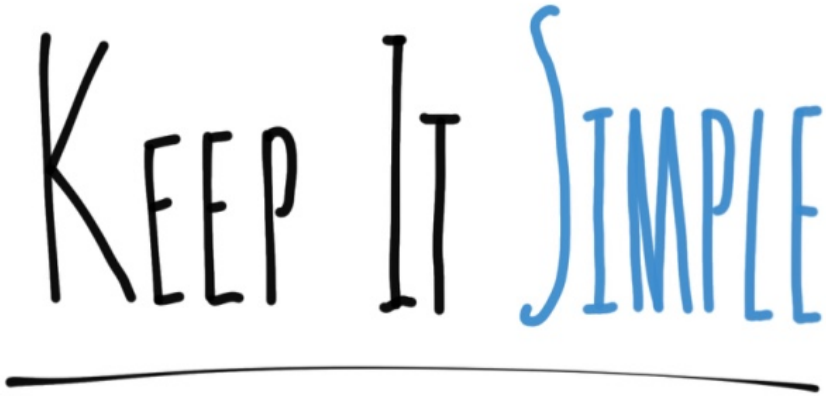
- Rapid innovation & delivery
- Developer happiness
- Attract & retain talent
- Performance



"Why Node.js is Becoming the Go-To Technology in the Enterprise" – nearform.com

Node.js exemplifies...

- Start with the simplest solution that works
- Do one thing, and do it well



Node.js Use Cases

- Single-page apps
- API server (REST, Hypermedia, etc.)
- Real-time, streaming
- WebSockets, push notifications
- Chat, IM, social media
- Dashboards
- Proxy service

Also Known As

The Internet

Node.js Use Cases

- Single-page apps
- API server (REST, Hypermedia, etc.)
- Real-time, streaming
- WebSockets, push notifications
- Chat, IM, social media
- Dashboards
- Proxy service

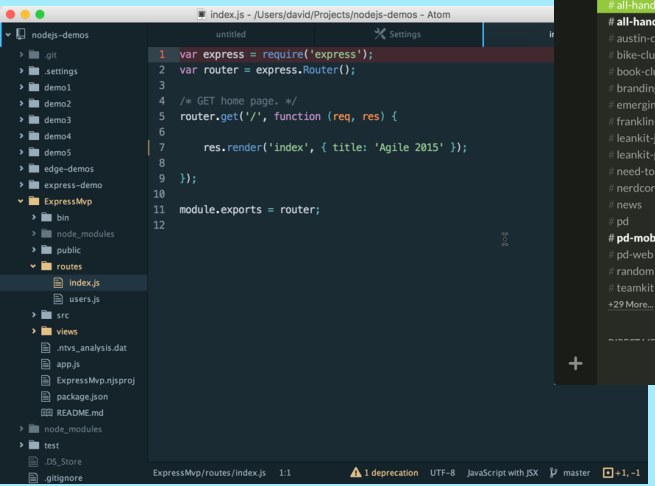
Also Known As

The Internet

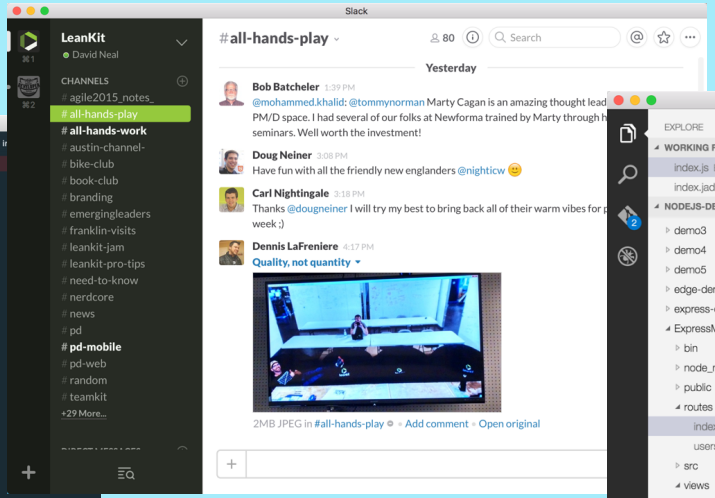
More than just backend

- Create Automation Scripts
- Create CLI tools using **nexe**
- Create Desktop Apps

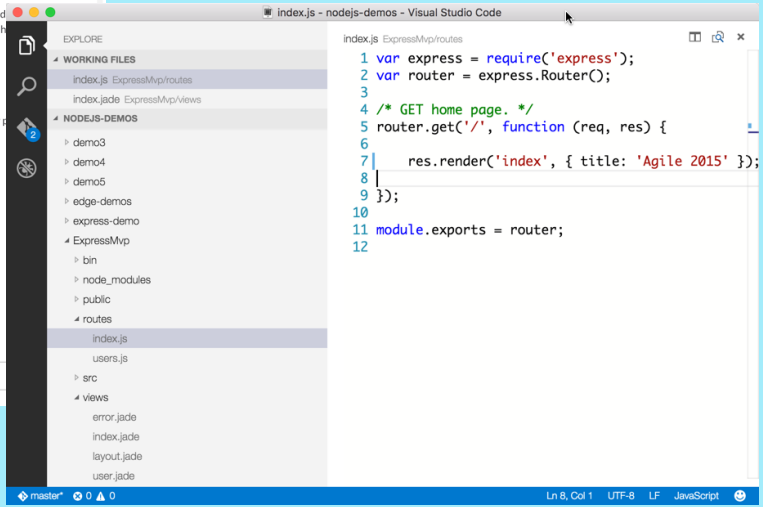
ELECTRON



Atom



Slack



Visual Studio Code

Fandango

- dramatically shorter development cycles
- micro-services architecture
- flexibility in deployment
- easily scalable infrastructure

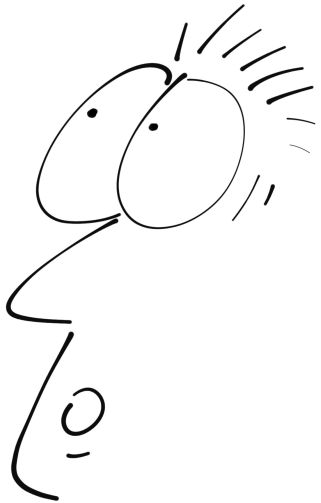
"Fandango Goes Live with Node.js" – nearform.com

- 2x faster development with fewer developers
- 33% fewer lines of code
- 40% fewer files
- 2x improvement requests/sec
- 35% decrease in avg response time

- Black Friday, 2013
- Mobile platform
- 200,000,000+ users
- 10 CPU cores, 28 GB RAM
- < 1% CPU utilization
- Deployed updates

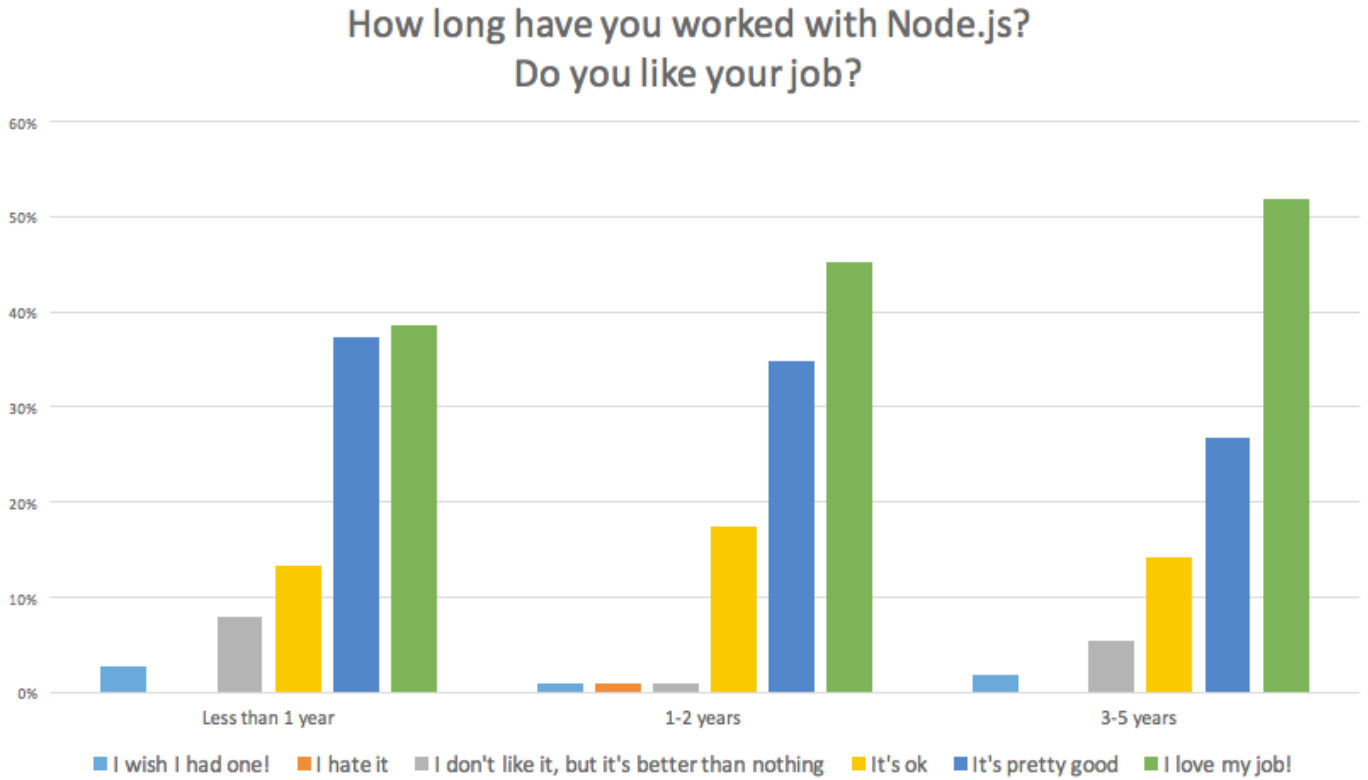
Who else is using Node.js?

- Dow Jones (WSJ)
- eBay
- Groupon
- LinkedIn
- IBM
- Redhat
- The New York Times
- Uber
- Yammer
- GoDaddy



DevOps

- Small footprint
- Cross-platform
- Event-driven
- OSS tools



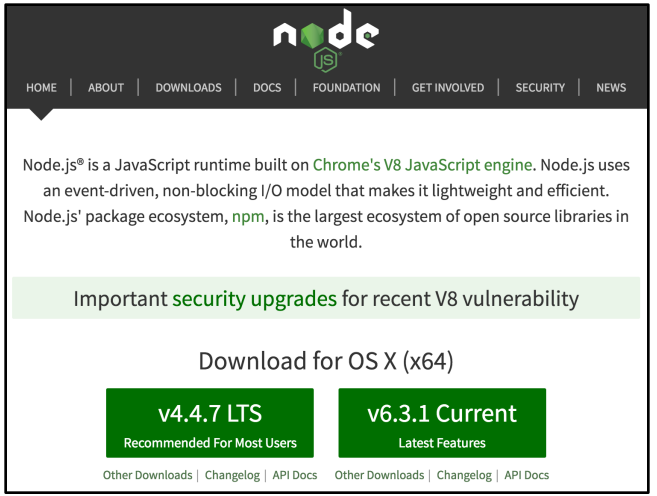
"What is Node.js used for: The 2015 Node.js Overview Report" – blog.risingstack.com

Installing Node.js

1. https://nodejs.org
2. Click a download link
3. Run installer

– OR –

Install using Chocolatey (https://chocolatey.org)




```
C:\> choco install nodejs.install
```

Installing Node.js

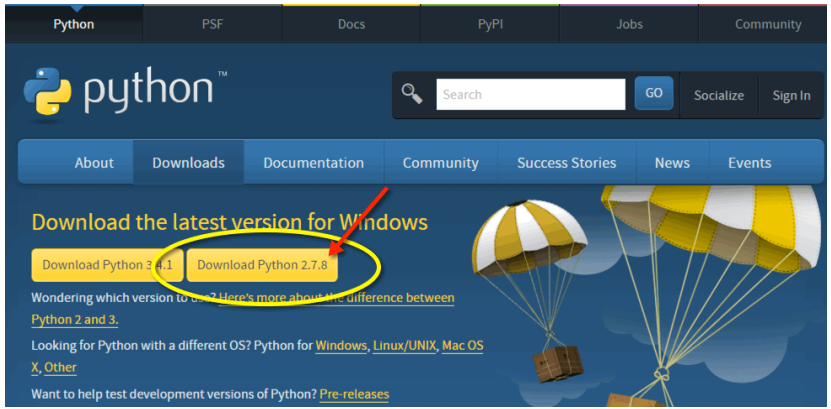
```
C:\> node -v
v6.3.1

C:\> npm -v
3.10.5
```

Dependencies

```
C:\> choco install python2
```

– OR –

Python 2.x (https://python.org/downloads/



Dependencies

```
C:\> choco install microsoft-build-tools
```

– OR –

Visual C++ Build Tools 2015

– OR –

Visual Studio Community 2015

Node.js Tools for Visual Studio

https://github.com/Microsoft/nodejstools

Minimum requirements:

- VS 2012 Pro
  or
  VS Community 2013/2015
- Latest VS updates
- VS + Node.js Tools Azure VM

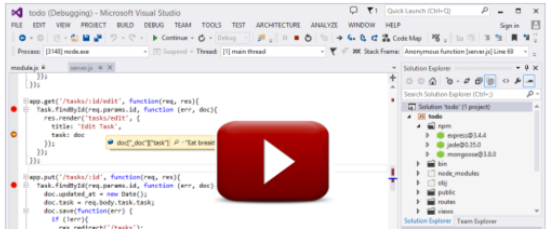
README.md

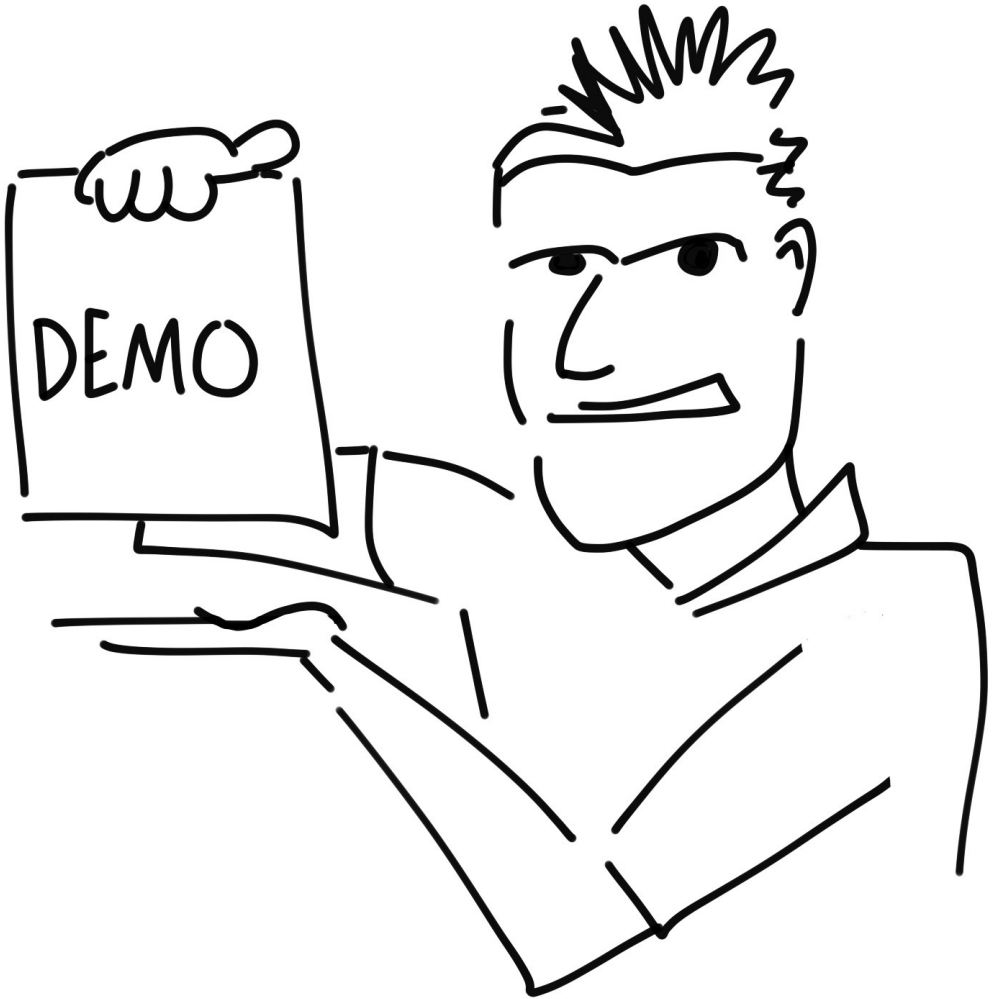
Node.js Tools for Visual Studio

NTVS is a free, open source plugin that turns Visual Studio into a Node.js IDE. It is designed, developed, and supported by Microsoft and the community.

build failing GITTER JOIN CHAT →

> Learn more and start developing Node.js applications in Visual Studio

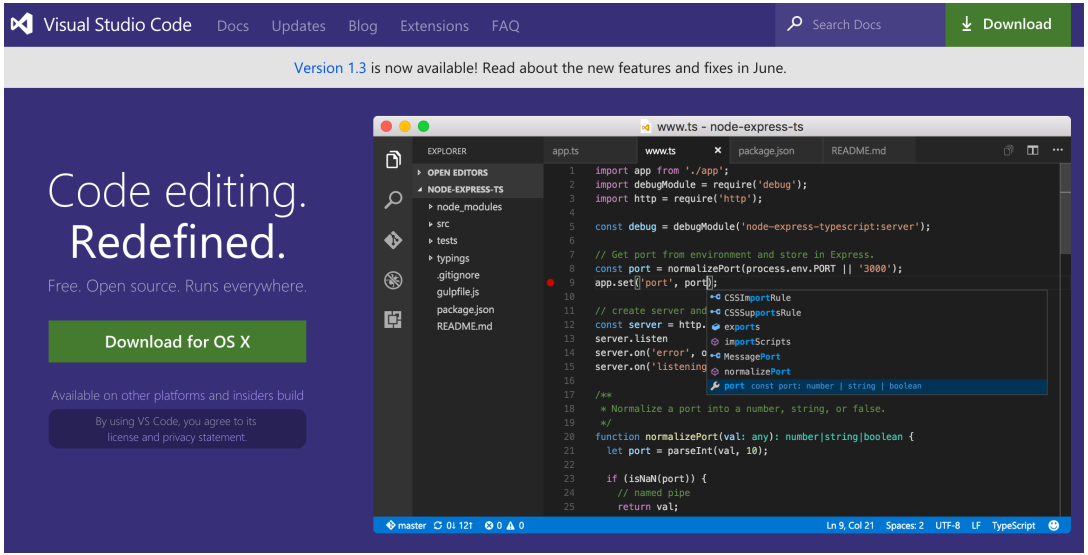



DEMO

Visual Studio Code

https://code.visualstudio.com/

- Linux, Mac OSX, and Windows
- code assistance
- debugging



```
C:\projects\myapp> npm init

C:\projects\myapp> npm install --save lodash
```

module-examples.js

```
1   var fs = require( "fs" ); // Built-in Node.js module
2
3   var express = require( "express" ); // Module installed via npm
4
5   var ApiService = require( "./app/ApiService" ); // Local module
6
7   var api = new ApiService( { port: 8888 } );
8
9   api.someMethod( "val1", 22, function( err, results ) {
10    if ( err ) {
11      // Do something
12    }
13     // Do stuff with results
14  } );
```

module-examples.js

```
1   var fs = require( "fs" ); // Built-in Node.js module
2
3   var express = require( "express" ); // Module installed via npm
4
5   var ApiService = require( "./app/ApiService" ); // Local module
6
7   var api = new ApiService( { port: 8888 } );
8
9   api.someMethod( "val1", 22, function( err, results ) {
10    if ( err ) {
11      // Do something
12    }
13     // Do stuff with results
14  } );
```

module-examples.js

```
1   var fs = require( "fs" ); // Built-in Node.js module
2
3   var express = require( "express" ); // Module installed via npm
4
5   var ApiService = require( "./app/ApiService" ); // Local module
6
7   var api = new ApiService( { port: 8888 } );
8
9   api.someMethod( "val1", 22, function( err, results ) {
10    if ( err ) {
11      // Do something
12    }
13     // Do stuff with results
14  } );
```

module-examples.js

```
1   var fs = require( "fs" ); // Built-in Node.js module
2
3   var express = require( "express" ); // Module installed via npm
4
5   var ApiService = require( "./app/ApiService" ); // Local module
6
7   var api = new ApiService( { port: 8888 } );
8
9   api.someMethod( "val1", 22, function( err, results ) {
10    if ( err ) {
11      // Do something
12    }
13     // Do stuff with results
14  } );
```

module-examples.js

```
1   var fs = require( "fs" ); // Built-in Node.js module
2
3   var express = require( "express" ); // Module installed via npm
4
5   var ApiService = require( "./app/ApiService" ); // Local module
6
7   var api = new ApiService( { port: 8888 } );
8
9   api.someMethod( "val1", 22, function( err, results ) {
10    if ( err ) {
11      // Do something
12    }
13     // Do stuff with results
14  } );
```

module-examples.js

```
1   var fs = require( "fs" ); // Built-in Node.js module
2
3   var express = require( "express" ); // Module installed via npm
4
5   var ApiService = require( "./app/ApiService" ); // Local module
6
7   var api = new ApiService( { port: 8888 } );
8
9   api.someMethod( "val1", 22, function( err, results ) {
10    if ( err ) {
11      // Do something
12    }
13     // Do stuff with results
14  } );
```

module-examples.js

```
1   var fs = require( "fs" ); // Built-in Node.js module
2
3   var express = require( "express" ); // Module installed via npm
4
5   var ApiService = require( "./app/ApiService" ); // Local module
6
7   var api = new ApiService( { port: 8888 } );
8
9   api.someMethod( "val1", 22, function( err, results ) {
10    if ( err ) {
11      // Do something
12    }
13     // Do stuff with results
14  } );
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temp-example.js

```
1   const t = require( "./temperature" );
2
3   const c = t.boilingCelcius;
4   const f = t.convertToFarenheit( c );
5
6   console.log( `${c} C is ${f} F` );
7   // output: 100 C is 212 F
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temp-example.js

```
1   const t = require( "./temperature" );
2
3   const c = t.boilingCelcius;
4   const f = t.convertToFarenheit( c );
5
6   console.log( `${c} C is ${f} F` );
7   // output: 100 C is 212 F
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temp-example.js

```
1   const t = require( "./temperature" );
2
3   const c = t.boilingCelcius;
4   const f = t.convertToFarenheit( c );
5
6   console.log( `${c} C is ${f} F` );
7   // output: 100 C is 212 F
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temp-example.js

```
1   const t = require( "./temperature" );
2
3   const c = t.boilingCelcius;
4   const f = t.convertToFarenheit( c );
5
6   console.log( `${c} C is ${f} F` );
7   // output: 100 C is 212 F
```

temperature.js

```
1   function convertToCelcius( f ) {
2       return ( f - 32 ) * ( 5 / 9 );
3   }
4
5   function convertToFarenheit( c ) {
6       return ( c * ( 9 / 5 ) ) + 32;
7   }
8
9   const temp = {
10      freezingCelcius: 0,
11      freezingFarenheit: 32,
12      boilingCelcius: 100,
13      boilingFarenheit: 212,
14      convertToCelcius,
15      convertToFarenheit
16  };
17
18  module.exports = temp;
```

temp-example.js

```
1   const t = require( "./temperature" );
2
3   const c = t.boilingCelcius;
4   const f = t.convertToFarenheit( c );
5
6   console.log( `${c} C is ${f} F` );
7   // output: 100 C is 212 F
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path ) {
4       const configData = fs.readFileSync( path, "utf8" );
5       const config = JSON.parse( configData );
6
7       const get = function( property, defaultValue ) { ···
10      const set = function( property, value ) { ···
13      const internalFunction = function() { ···
16      const save = function( callback ) { ···
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path ) {
4       const configData = fs.readFileSync( path, "utf8" );
5       const config = JSON.parse( configData );
6
7       const get = function( property, defaultValue ) { ···
10      const set = function( property, value ) { ···
13      const internalFunction = function() { ···
16      const save = function( callback ) { ···
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path ) {
4       const configData = fs.readFileSync( path, "utf8" );
5       const config = JSON.parse( configData );
6
7       const get = function( property, defaultValue ) { ···
10      const set = function( property, value ) { ···
13      const internalFunction = function() { ···
16      const save = function( callback ) { ···
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path ) {
4       const configData = fs.readFileSync( path, "utf8" );
5       const config = JSON.parse( configData );
6
7       const get = function( property, defaultValue ) { ···
10      const set = function( property, value ) { ···
13      const internalFunction = function() { ···
16      const save = function( callback ) { ···
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path ) {
4       const configData = fs.readFileSync( path, "utf8" );
5       const config = JSON.parse( configData );
6
7       const get = function( property, defaultValue ) { ···
10      const set = function( property, value ) { ···
13      const internalFunction = function() { ···
16      const save = function( callback ) { ···
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path ) {
4       const configData = fs.readFileSync( path, "utf8" );
5       const config = JSON.parse( configData );
6
7       const get = function( property, defaultValue ) { ···
10      const set = function( property, value ) { ···
13      const internalFunction = function() { ···
16      const save = function( callback ) { ···
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path ) {
4       const configData = fs.readFileSync( path, "utf8" );
5       const config = JSON.parse( configData );
6
7       const get = function( property, defaultValue ) { ···
10      const set = function( property, value ) { ···
13      const internalFunction = function() { ···
16      const save = function( callback ) { ···
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path )
4       const configData = fs.readFileSyn
5       const config = JSON.parse( config
6
7       const get = function( property, c
10      const set = function( property, v
13      const internalFunction = function
16      const save = function( callback )
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

cfg-example.js

```
1   const Config = require( "./config" );
2   const appConfig = new Config( "./app.json" );
3
4   const mw = appConfig.get( "minWidth", 100 );
5
6   console.log( mw );
7   // output: 250
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path )
4       const configData = fs.readFileSyn
5       const config = JSON.parse( config
6
7       const get = function( property, c
10      const set = function( property, v
13      const internalFunction = function
16      const save = function( callback )
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

cfg-example.js

```
1   const Config = require( "./config" );
2   const appConfig = new Config( "./app.json" );
3
4   const mw = appConfig.get( "minWidth", 100 );
5
6   console.log( mw );
7   // output: 250
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path )
4       const configData = fs.readFileSyn
5       const config = JSON.parse( config
6
7       const get = function( property, c
10      const set = function( property, v
13      const internalFunction = function
16      const save = function( callback )
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

cfg-example.js

```
1   const Config = require( "./config" );
2   const appConfig = new Config( "./app.json" );
3
4   const mw = appConfig.get( "minWidth", 100 );
5
6   console.log( mw );
7   // output: 250
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path )
4       const configData = fs.readFileSyn
5       const config = JSON.parse( config
6
7       const get = function( property, c
10      const set = function( property, v
13      const internalFunction = function
16      const save = function( callback )
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

cfg-example.js

```
1   const Config = require( "./config" );
2   const appConfig = new Config( "./app.json" );
3
4   const mw = appConfig.get( "minWidth", 100 );
5
6   console.log( mw );
7   // output: 250
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path )
4       const configData = fs.readFileSyn
5       const config = JSON.parse( config
6
7       const get = function( property, c
10      const set = function( property, v
13      const internalFunction = function
16      const save = function( callback )
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

cfg-example.js

```
1   const Config = require( "./config" );
2   const appConfig = new Config( "./app.json" );
3
4   const mw = appConfig.get( "minWidth", 100 );
5
6   console.log( mw );
7   // output: 250
```

config.js

```
1   const fs = require( "fs" );
2
3   const configModule = function( path )
4       const configData = fs.readFileSyn
5       const config = JSON.parse( config
6
7       const get = function( property, c
10      const set = function( property, v
13      const internalFunction = function
16      const save = function( callback )
21      return {
22          get,
23          set,
24          save
25      };
26  };
27
28  module.exports = configModule;
```

cfg-example.js

```
1   const Config = require( "./config" );
2   const appConfig = new Config( "./app.json" );
3
4   const mw = appConfig.get( "minWidth", 100 );
5
6   console.log( mw );
7   // output: 250
```

app.json

```
1   {
2       "minWidth": 250,
3       "maxWidth": 900,
4       "minHeight": 250,
5       "maxHeight": 1200
6   }
```

sql-example.js

```
1   const sql = require( "seriate" );
2
3   // Read from environment variables or local config file
4   const config = { server: "127.0.0.1", user: "user1", password: "mypass", database: "mydb" };
5   sql.setDefaultConfig( config );
6
7   sql.execute( {
8       query: "SELECT userId, email FROM accounts"
9   } ).then( function( results ) {
10      // Do something with the results
11      console.log( results );
12  } ).catch( function( err ) {
13      console.log( "Something bad happened:", err );
14  } );
```

sql-example.js

```
1   const sql = require( "seriate" );
2
3   // Read from environment variables or local config file
4   const config = { server: "127.0.0.1", user: "user1", password: "mypass", database: "mydb" };
5   sql.setDefaultConfig( config );
6
7   sql.execute( {
8       query: "SELECT userId, email FROM accounts"
9   } ).then( function( results ) {
10      // Do something with the results
11      console.log( results );
12  } ).catch( function( err ) {
13      console.log( "Something bad happened:", err );
14  } );
```

sql-example.js

```
1   const sql = require( "seriate" );
2
3   // Read from environment variables or local config file
4   const config = { server: "127.0.0.1", user: "user1", password: "mypass", database: "mydb" };
5   sql.setDefaultConfig( config );
6
7   sql.execute( {
8       query: "SELECT userId, email FROM accounts"
9   } ).then( function( results ) {
10      // Do something with the results
11      console.log( results );
12  } ).catch( function( err ) {
13      console.log( "Something bad happened:", err );
14  } );
```

sql-example.js

```
1   const sql = require( "seriate" );
2
3   // Read from environment variables or local config file
4   const config = { server: "127.0.0.1", user: "user1", password: "mypass", database: "mydb" };
5   sql.setDefaultConfig( config );
6
7   sql.execute( {
8       query: "SELECT userId, email FROM accounts"
9   } ).then( function( results ) {
10      // Do something with the results
11      console.log( results );
12  } ).catch( function( err ) {
13      console.log( "Something bad happened:", err );
14  } );
```

sql-example.js

```
1   const sql = require( "seriate" );
2
3   // Read from environment variables or local config file
4   const config = { server: "127.0.0.1", user: "user1", password: "mypass", database: "mydb" };
5   sql.setDefaultConfig( config );
6
7   sql.execute( {
8       query: "SELECT userId, email FROM accounts"
9   } ).then( function( results ) {
10      // Do something with the results
11      console.log( results );
12  } ).catch( function( err ) {
13      console.log( "Something bad happened:", err );
14  } );
```

Deploying

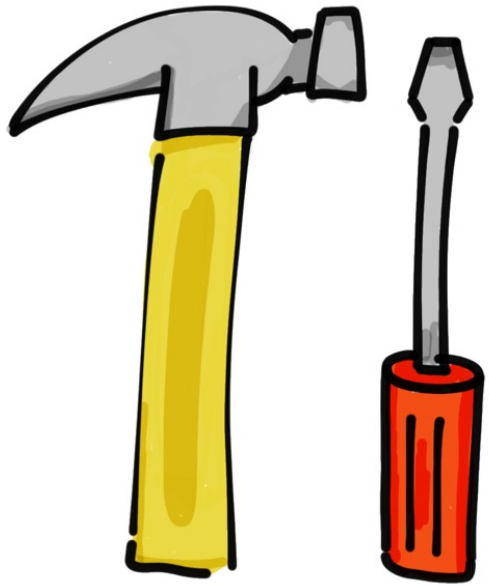
- Windows
  - `iisnode` for web apps
  - `winser` for services
- Linux – `forever`

Hosting Node.js on Azure

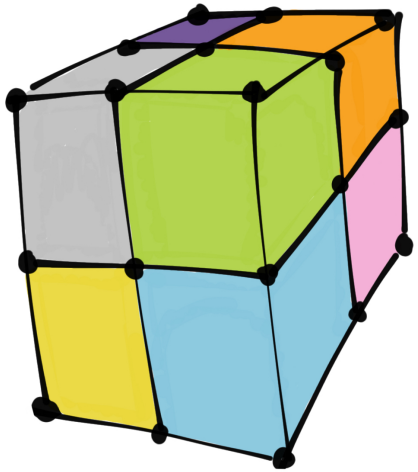
- New Node JS Empty Site
- Connect to repository
- reverentgeek.com/hosting-node-js-on-microsoft-azure/

Recommended toolbox

Package	What it do, yo
lodash	JavaScript utilities
when	JavaScript promise library
async	async/parallel execution
request (or rest)	http client
gulp	build engine, test runner
socket.io	sockets, real-time
node-inspector	Debugging
mocha	test framework
chai	TDD/BDD assertion library
sinon	spies, stubs, mocks



Node frameworks



MVC

- Express
- hapi
- Meteor
- Sails

API

- Restify
- LoopBack
- Autohost + Hyped

nodeframework.com
nodewebmodules.com

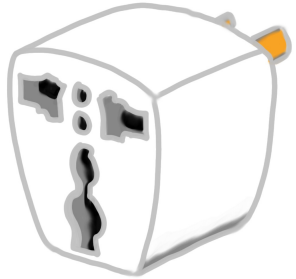
- Run .NET in-process
- ...including F#, ADO.NET, Python, and Powershell
- Execute inline code, files, or assemblies
- Alternative to writing native modules in C
- .NET 4.5 or Mono 3.1

What can Edge.js do?

- Leverage existing .NET investment
- TFS, SharePoint, Exchange, etc.
- Active Directory
- Hardware (e.g. camera, microphone, printer, win32)
- Video encoding, or other CPU-intensive work
- Powershell

Node.js Integration Strategies

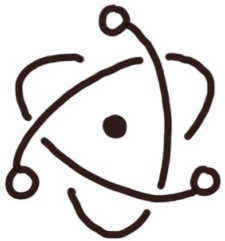
- Node.js as proxy
- Edge.js for .NET
- `request` module to call APIs
- Messaging (e.g. RabbitMQ, Azure Service Bus)



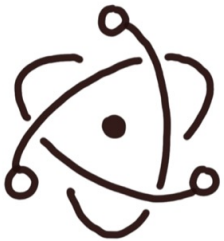

You Don't Need
Permission
To Be
Awesome!

CROSS-PLATFORM DESKTOP APPS

* WITH *



ELECTRON



Thank You!

DAVID NEAL
@ReverentGeek
david@reverentgeek.com

Demos + LOTS of Resources
bit.ly/node-demos