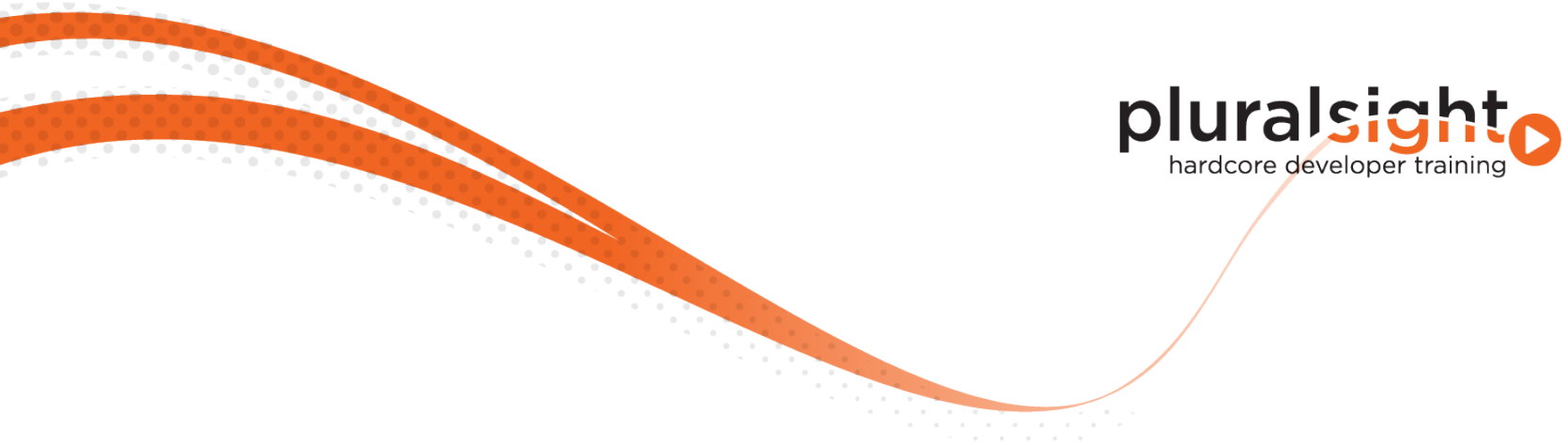
Getting Started with Node.js

Paul O'Fallon
@paulofallon

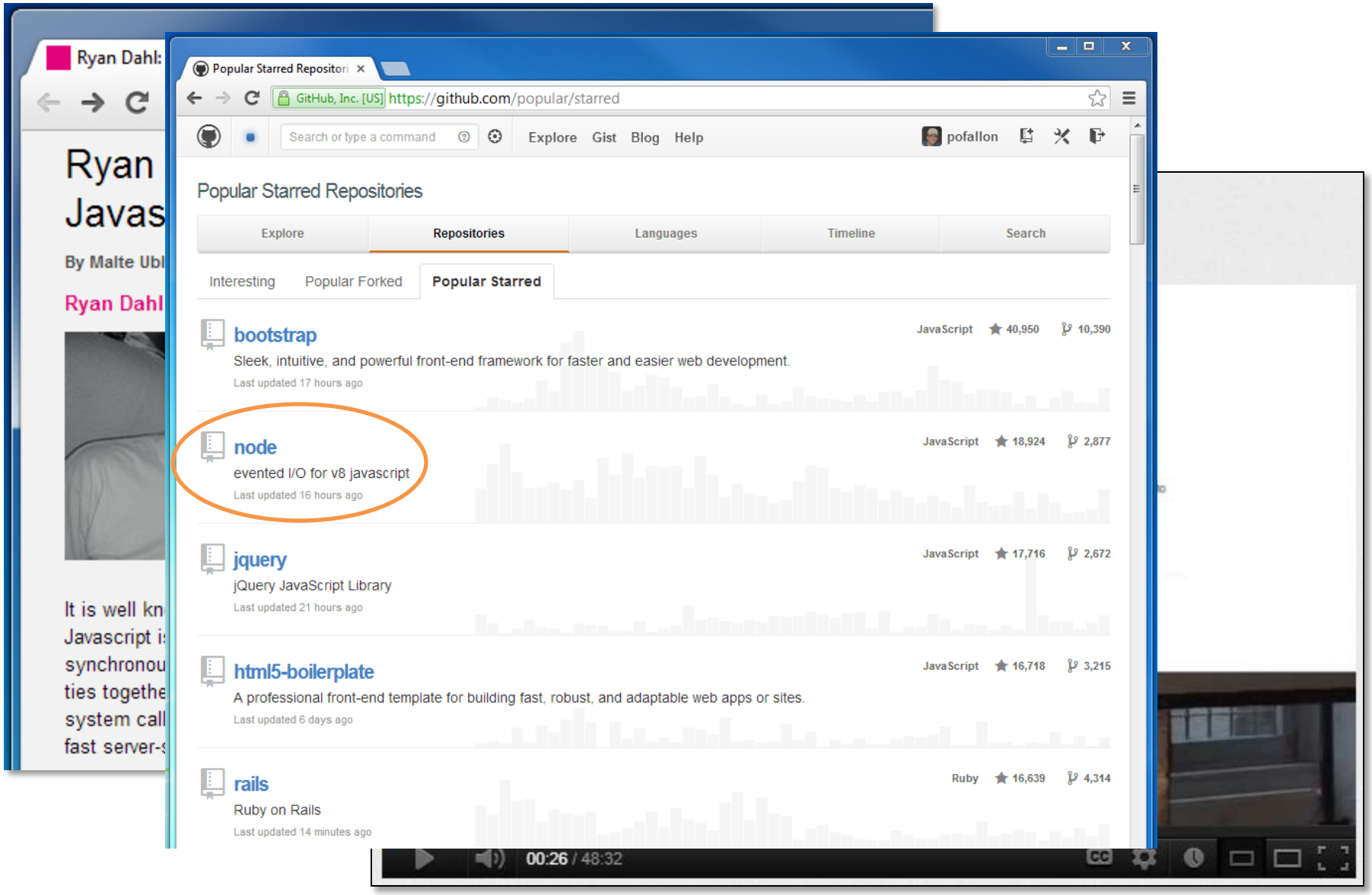


pluralsight
hardcore developer training

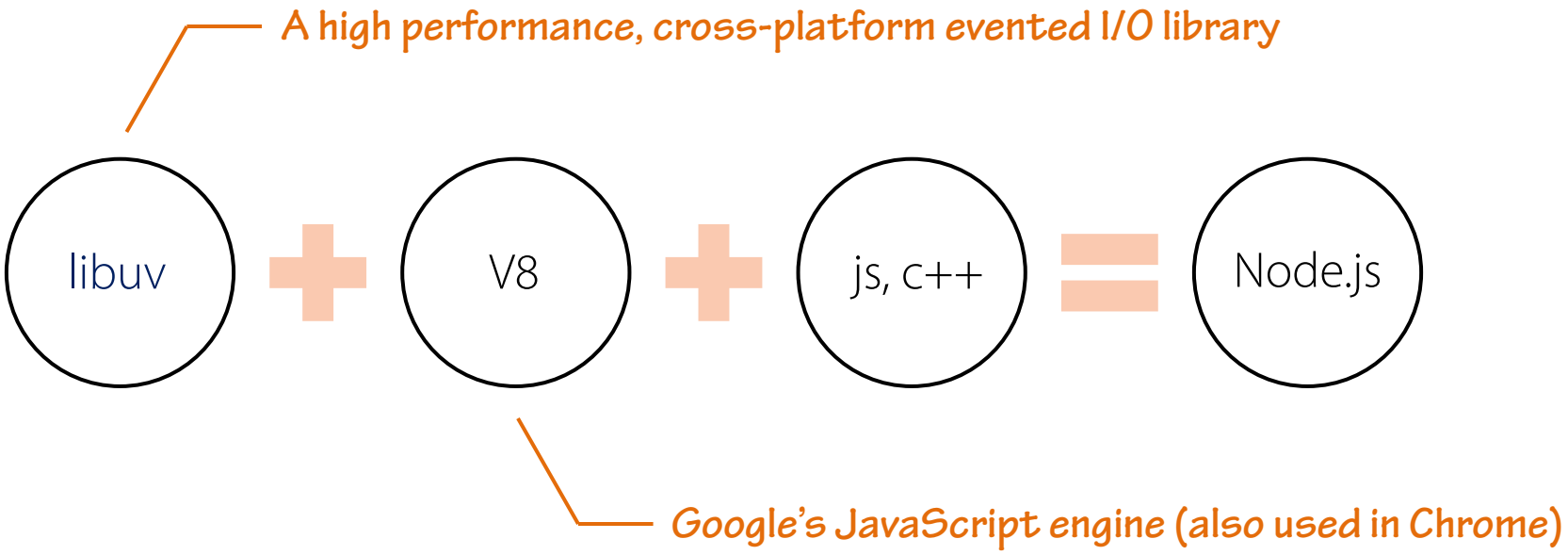
Outline

- An overview of Node.js
- Building and installing Node.js
- Developing for Node with Cloud9 IDE
- An introduction to Node's event loop
- Writing code with callbacks

Node.js background



Node.js Building Blocks



Getting Node.js

http://nodejs.org/download/

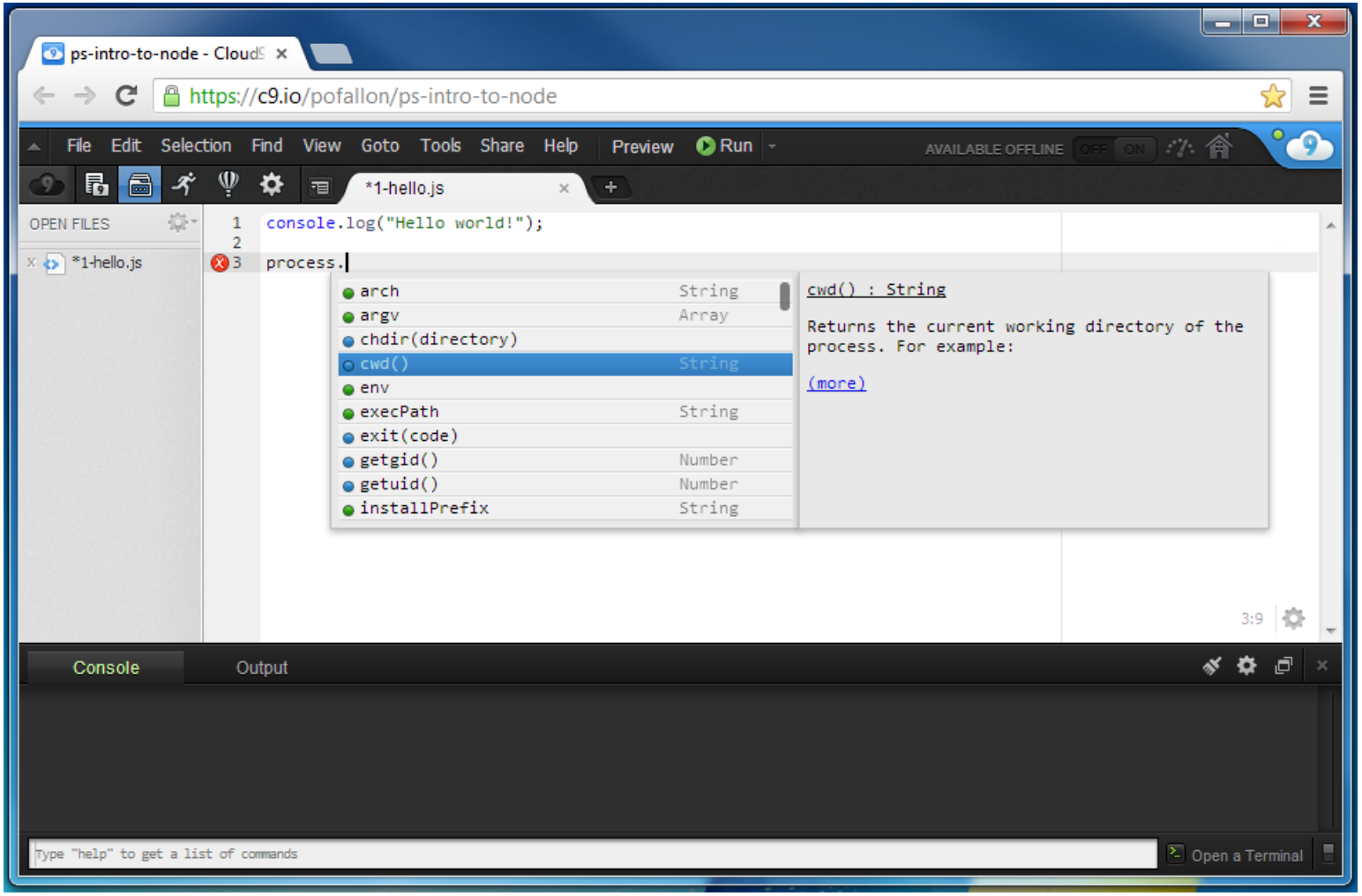
- **Installers available for Windows & Mac OS X**
- **Binaries available for Windows, Mac, Linux and SunOS**
  - Also available via many Linux package managers
- **Source code also available**

Manage multiple versions with 'nvm' (https://github.com/creationix/nvm)

- `git clone git://github.com/creationix/nvm.git ~/nvm`
- `. ~/nvm/nvm.sh`
- `nvm install 0.8.14`
- `nvm use 0.6.19`
- `nvm alias default 0.8.14`


pluralsight
hardcore developer training

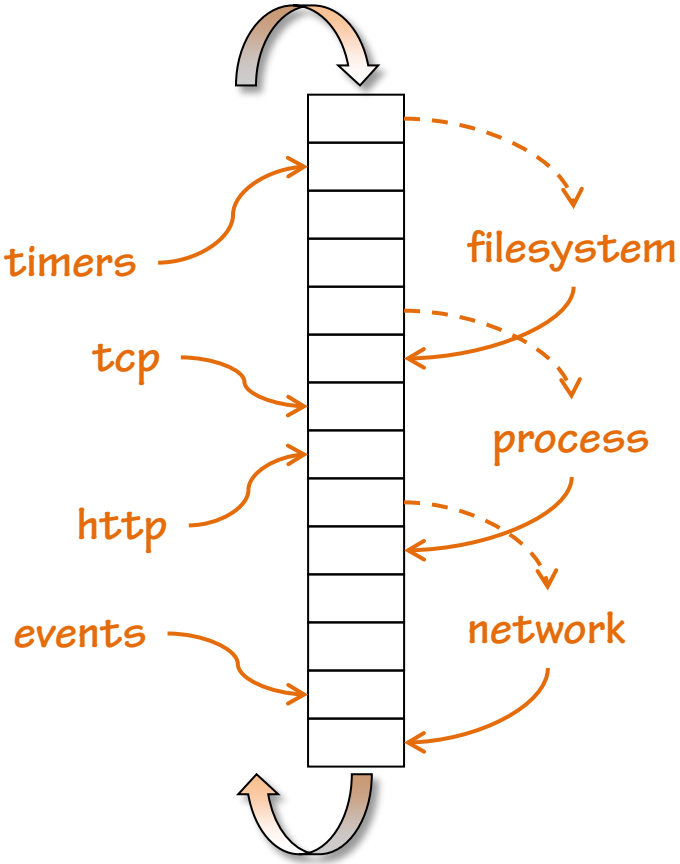
Cloud9 IDE



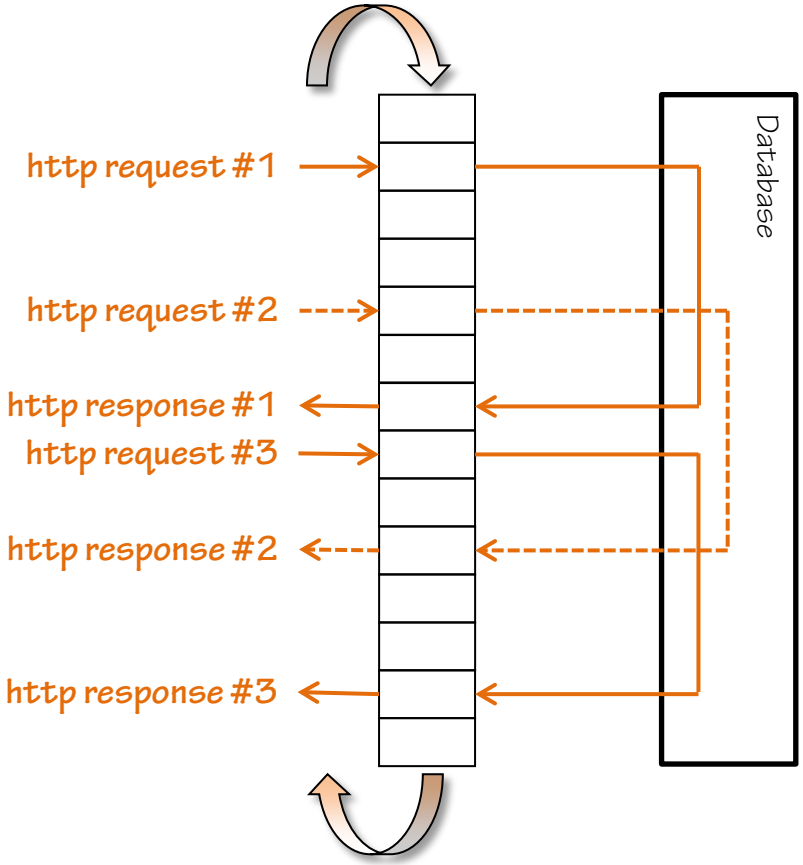
http://c9.io


pluralsight
hardcore developer training

Node's Event Loop



What does this mean in practice?




pluralsight
hardcore developer training

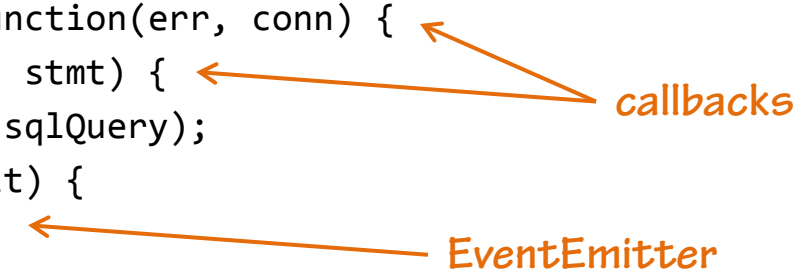
Writing asynchronous code is different

A typical approach

```
var conn = getDbConnection(connectionString);
var stmt = conn.createStatement();
var results = stmt.executeQuery(sqlQuery);
for (var i=0; i<results.length; i++) {
  // print results[i];
}
```

An asynchronous, “non-blocking” approach

```
getDbConnection(connectionString, function(err, conn) {
  conn.createStatement(function(err, stmt) {
    var results = stmt.executeQuery(sqlQuery);
    results.on('row', function(result) {
      // print result
    });
  });
});
```



Coding for asynchrony with callbacks

Asynchronous functions with callbacks

② Error is first parameter to callback function

```
var handleResults = function(error, results) {
  // if error is undefined…
  // do something with the results
}

getStuff(inputParam, handleResults);
```

① Callback is last parameter in async function call

Anonymous Functions and Closures

For simple callbacks, anonymous functions are more common

```
getStuff(inputParam, function(error, results) {
  // if error is undefined…
  // do something with the results
});
```

… and closures are your friend!

```
someOtherFunction(function(err, stuffToGet) {
  var foo = 23;
  getStuff(stuffToGet, function(error, results) {
    // if error is undefined…
    // do something with the results (and foo)
  });
});
```


pluralsight
hardcore developer training

Too much of a good thing…

Beware of the “Christmas tree” effect!

```
asyncFunction1(inputParam, function(err, results1) {
  asyncFunction2(results1, function (err, results2) {
    asyncFunction3(results2, function (err, results3) {
      asyncFunction4(results3, function (err, results4) {
        asyncFunction5(results4, function (err, results5) {
          // and so on…
        });
      });
    });
  });
});
```

Conclusion

- Overview of Node.js and its beginnings
- Installing Node.js and using nvm to manage versions
- Introduction to Cloud9 IDE
- The importance of Node's event loop and non-blocking I/O
- Writing asynchronous code using callbacks


pluralsight
hardcore developer training

References

- Ryan Dahl's original JSConf.eu presentation (YouTube): http://www.youtube.com/watch?v=ztspvPYybIY
- An introduction to libuv - http://nikhilm.github.com/uvbook/introduction.html
- V8 JavaScript Engine (Google Project Hosting): http://code.google.com/p/v8/
- Installing Node via a package manager: https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager
- How to JavaScript closures work? http://stackoverflow.com/questions/111102/how-do-javascript-closures-work