

Node.JS



Server side using Javascript

How are we doing?

General

- How the web works?
- Git
 - GitHub
 - Bitbucket
 - SourceTree
- Deployment
- Chrome dev tools
- Bonus: latest trends

Client side

- HTML5: What's new?
- CSS3 (Advanced selectors)
- Web layout with Flexbox
- Javascript & jQuery

Server side

- NodeJS
- NPM (and such)
- Express
- MongoDB
- MongoChef

What is Node.JS?

Open Source
server side
runtime environment

Truly cross platform

Uses JavaScript as its
language!

Our stack



+

Web Development
Framework

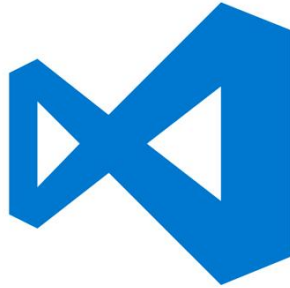


+

express

Unopinionated
&
minimalist

Pick your editor



Let's get started

1. Install Node.JS
 2. New repository (reminder)
 3. Create & Run *helloWorld.js*
 4. Use NPM to get a package
 5. What is .gitignore?
 6. Use an NPM package
 7. Commit & Push
-

What is NPM?



Package Management
for Node.js

+

Easy Updates and
Version Management



Common NPM commands

- **npm init** - Creates a package.json file, which describes your project
- **npm install PACKAGE_NAME --save** - installs a package and adds it to package.json
- **npm install** - install all packages that are required in package.json

One more thing..

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
        });
    });
});
```

callbacks

EventEmitter

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)  
  });  
});
```

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



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...  
        });  
      });  
    });  
  });  
});  
});  
});
```