

node.js

evented I/O for V8 JavaScript

what is node?

- Server-side JavaScript done right.
- Runs on V8
- An environment for developing high-performance web services
- Evented TCP stack
- Not a framework

why node?

- Web applications spend most of their time doing I/O
- JavaScript is the language of the web

V8

V8

- Google's open source JavaScript engine.
- Developed by Lars Bak.
- Fast: compiles JavaScript to machine code.
- Implements most of ECMAScript 5.

ECMAScript 5

ECMAScript 5

Safe prototype extension

```
Object.defineProperty(Object.prototype, "forEach", {  
  value: function (callback) {  
    var keys = Object.keys(this);  
  
    for (var i = 0, key; i < keys.length; i++) {  
      key = keys[i];  
      callback.call(this, key, this[key]);  
    }  
  }  
});
```

ECMAScript 5

Access to the hidden prototypes

```
Object.getPrototypeOf([ ]) // Array  
[ ].__proto__                // Array  
[ ].__proto__.__proto__     // Object
```

ECMAScript 5

Basic prototypal inheritance

```
var o = Object.create({ foo: 42 });
```

```
o.bar = "bah";
```

```
Object.keys(o) // ["bar"]
```

```
o.foo // 42
```

```
o.__proto__ // { foo: 42 }
```

node.js

node.js

Event-driven programming

- **Asynchronous I/O**
- **Callbacks**

node.js

Common.js module system

```
var sys = require("sys");
```

```
sys.puts("hello world");
```

node.js

Common.js module system

```
require.paths    // ["/lib", ...]  
__dirname       // this dirname  
__filename      // this filename
```

node.js

Simple HTTP server

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

http.createServer( function (request, response) {
    response.writeHead(200, {
        'Content-Type': 'text/plain'
    });
    response.end( 'Hello World\n' );
}).listen(8000);
```

Event-driven programming

```
setTimeout(function () {  
    // Do something after 1 second  
}, 1000);
```

```
process.nextTick(function () {  
    // Do something asynchronously  
});
```

Async error handling

```
process.addListener( 'uncaughtException',  
                    function (err) {  
                        // Handle exception  
                    });
```

Async signal handling

```
process.addListener( 'SIGINT', function (err) {  
    // Handle Ctrl-C  
});
```

modules

file-system module

```
require( 'fs' );
```

modules

fs

- one-to-one mapping with unix commands
- most functions have a synchronous version

Asynchronous file stat

```
fs.stat("path/to/file", function (err, res) {  
  if (res) {  
    // Handle success  
  } else {  
    // Handle error  
  }  
});
```

Synchronous file stat

```
var res = fs.statSync("path/to/file");  
  
if (res) {  
    // Handle success  
} else {  
    // Handle error  
}
```

<http://nodejs.org>

@cloudhead

Alexis Sellier

<http://github.com/cloudhead/node-intro>