INTRO TO NODE.JS

- LEVEL ONE -

WHAT IS NODE.JS?

Allows you to build scalable network applications using JavaScript on the server-side.

Node.js

V8 JavaScript Runtime

It's fast because it's mostly C code

WHAT COULD YOU BUILD?

- Websocket Server *Like a chat server*
- Fast File Upload Client
- Ad Server
- Any Real-Time Data Apps

WHAT IS NODE.JS NOT?

- A Web Framework
- For Beginners *It's very low level*
- Multi-threaded

You can think of it as a single threaded server

OBJECTIVE: PRINT FILE CONTENTS

- Blocking Code

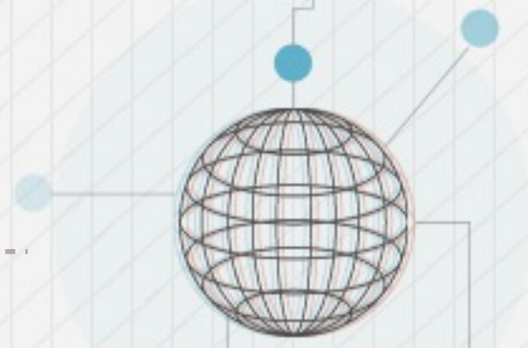
```
Read file from Filesystem, set equal to “contents”
Print contents
Do something else
```

- Non-Blocking Code

```
Read file from Filesystem
  whenever you’re complete, print the contents
Do Something else
```

This is a “Callback”

BLOCKING VS NON-BLOCKING



- Blocking Code

```
var contents = fs.readFileSync('/etc/hosts');
console.log(contents);
console.log('Doing something else');
```

Stop process until complete

- Non-Blocking Code

```
fs.readFile('/etc/hosts', function(err, contents) {
  console.log(contents);
});
console.log('Doing something else');
```

CALLBACK ALTERNATE SYNTAX

```
fs.readFile('/etc/hosts', function(err, contents) {
  console.log(contents);
});
```

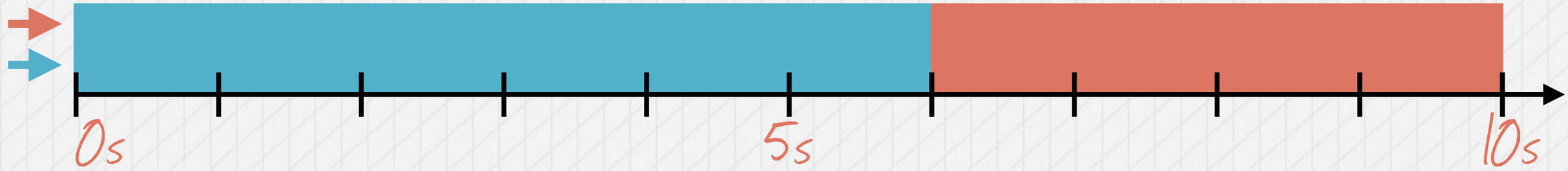
Same as

```
var callback = function(err, contents) {
  console.log(contents);
}

fs.readFile('/etc/hosts', callback);
```

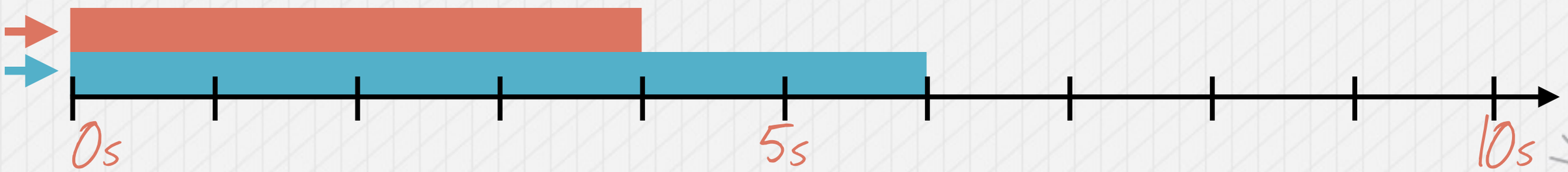
BLOCKING VS NON-BLOCKING

```
var callback = function(err, contents) {
  console.log(contents);
}

fs.readFile('/etc/hosts', callback);
fs.readFile('/etc/inetcfg', callback);
```

blocking



non-blocking



NODE.JS HELLO DOG

hello.js

```
var http = require('http');  How we require modules

http.createServer(function(request, response) {
  response.writeHead(200);   Status code in header
  response.write("Hello, this is dog.");  Response body
  response.end(); Close the connection
}).listen(8080, function(){  Listen for connections on this port
  console.log('Listening on port 8080...');
});
```

```
$ node hello.js   Run the server
```

----> Listening on port 8080...

```
$ curl http://localhost:8080
```

----> Hello, this is dog.

THE EVENT LOOP

```
var http = require('http');
http.createServer(function(request, response) {
  ...
}).listen(8080, function(){
  console.log('Listening on port 8080...');
});
```

Starts the Event Loop when finished

Run the Callback

Known Events

request

WHY JAVASCRIPT?

"JavaScript has certain characteristics that make it very different than other dynamic languages, namely that it has no concept of threads. Its model of concurrency is completely based around events."

- Ryan Dahl

THE EVENT LOOP

Event Queue

- close
- request

Known Events

- request
- connection
- close

Events processed one at a time

WITH LONG RUNNING PROCESS

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

http.createServer(function(request, response) {
  response.writeHead(200);
  response.write("Dog is running.");
  setTimeout(function(){    Represent long running process
    response.write("Dog is done.");
    response.end();
  }, 5000);  5000ms = 5 seconds
}).listen(8080);
```

TWO CALLBACKS HERE

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

http.createServer(function(request, response) {
  response.writeHead(200);
  response.write("Dog is running.");
  setTimeout(function(){
    response.write("Dog is done.");
    response.end();
  }, 5000);
}).listen(8080);
```

request

timeout

TWO CALLBACKS TIMELINE

- Request comes in, triggers request event
  - Request Callback executes
    - setTimeout registered
      - Request comes in, triggers request event
        - Request Callback executes
        - setTimeout registered
      - triggers setTimeout event
      - setTimeout Callback executes
      - triggers setTimeout event
      - setTimeout Callback

request

timeout

0s

5s

10s

WITH BLOCKING TIMELINE

Request comes in, triggers request event

Request Callback executes

setTimeout executed

Request comes in, waits for server

triggers setTimeout event

setTimeout Callback executed

Request comes in

Request Callback executes

Wasted Time

0s

5s

10s

TYPICAL BLOCKING THINGS

- Calls out to web services
- Reads/Writes on the Database
- Calls to extensions