Node.js

Embedded web server

Topics

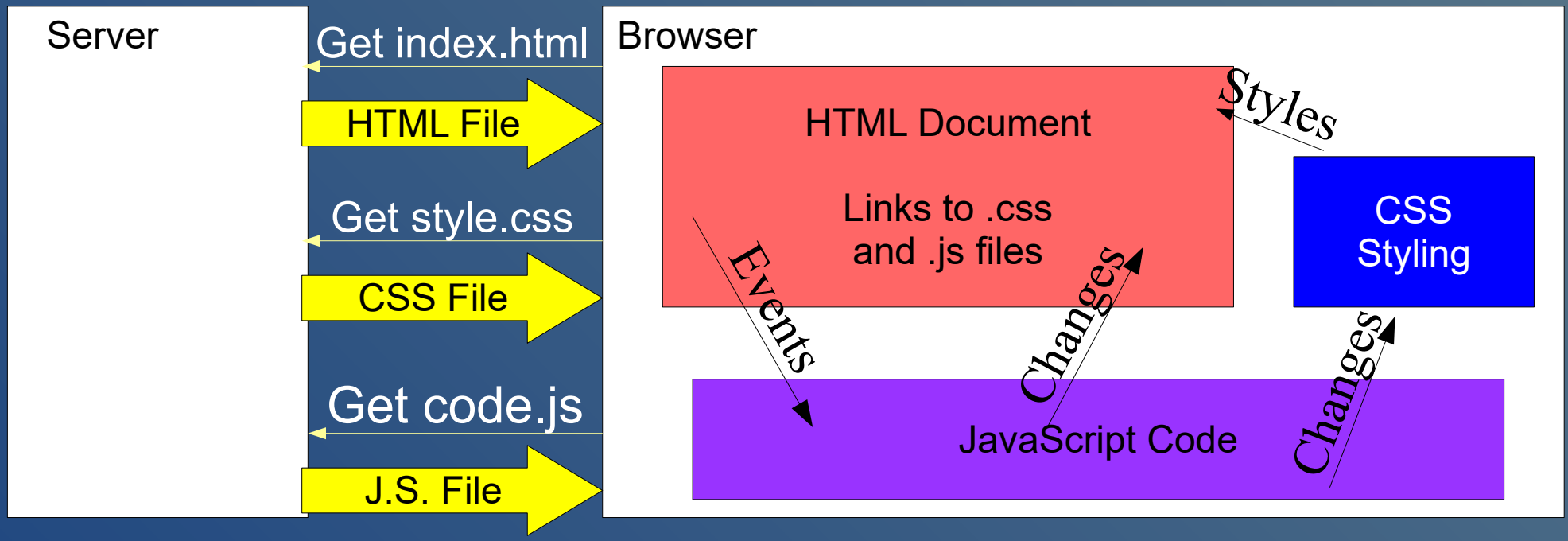
1) How to build a static web pages: .html, .css, .js?
2) How to serve static pages with Node.js?
3) How to create dynamic content via WebSocket?
4) How to connect Node.js to C program?

Static Client Pages

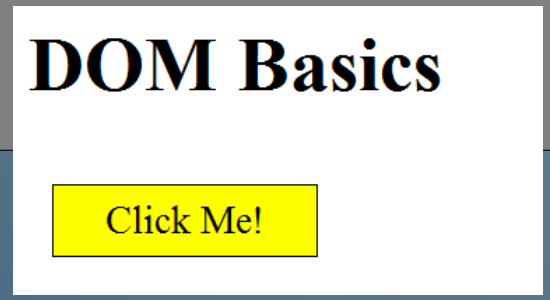
HTML, CSS, and JavaScript

Static Client Content

- Static content is stored in files on a server and sent to the client on demand.
  - File content does not dynamically change.



Static Files



XHTML: index.html

```
<html>
<head>
	<title>DOM Basics</title>
	<link rel="stylesheet" href="style.css"
		type="text/css"/>
</head>

<body>
	<h1>DOM Basics</h1>
	<div id="daBox" onclick="yaClickedBox()">
		Click Me!
	</div>
	<script type="text/javascript" src="code.js">
	</script>
</body>
</html>
```

CSS: style.css

```
#daBox {
	border: thin black solid;
	background-color: yellow;
	margin: 10px;
	padding: 5px;
	float: left;
	width: 100px;
	text-align: center;
}
```

JavaScript: code.js

```
function yaClickedBox() {
	// Your Code Here...
}
```

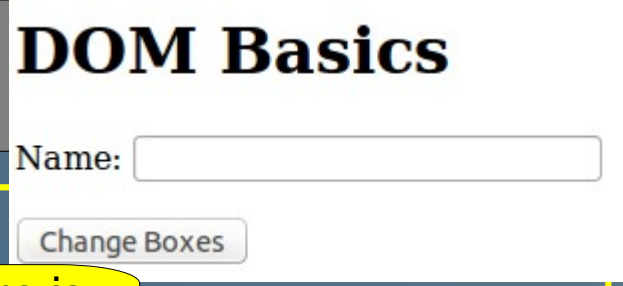
JavaScript Basics

- JavaScript:
  - case sensitive, dynamically typed
  - ; at end of statements optional
- String: "Hello World" same as 'Hello World'
- Variables
  - ```
    var str = "123";
    var x = str.length;
    var y = Number(str);          // Convert string to number
    str = 5;                      // Change type
    ```
  - Can create a variable without declaration:
    ```
    terribleIdea = 43;            // Why bad?
    ```
    Make this illegal by placing this at top of file:
    ```
    "use strict";   // quotes included!
    ```
    ```

DOM

- Client-side JavaScript runs in the browser
  - i.e., It's runtime environment is the browser.
  - Can interact with HTML and CSS that make up the currently loaded web page ("document").
  - Called the.. Document Object Model (DOM).
  - ```
    function changeBox() {
        // Change HTML code "inside" the the div "box":
        $('#daBox').html("Hello World");
    }
    ```


    ```

jQuery

- jQuery
  - A client-side JavaScript library to simplify interacting with the browser (DOM).
- Use in JavaScript:
  - $('#myStuff'): gets the.. DOM object of ID “myStuff”
    - In HTML: .....
    - In JavaScript (change contents):
      $('#myStuff').html("Hello world!");
  - $(' '): Create a new DOM object.
    - Example: Add text to a new div:
      var block = $(' ').text('Hello world!');

Form Example




```
<body>
<form action="">
    <h1>DOM Basics</h1>
    <p>Name:
        <input type="text" id="nameId"/>
    </p>
    <p>
        <input type="button" id="changeBtn"
            value="Change Boxes"/>
    </p>
    <div id="box1">Box 1</div>

    <script
        src='http://code.jquery.com/jquery-1.11.1.min.js'
        type='text/javascript'></script>
    <script type="text/javascript"
        src="javascripts/code.js"></script>
</form>
</body>
</html>
```

<form> wraps all input elements.

Text entry box

Clickable button.

JQuery library

Our code

```
"use strict";

$(document).ready(function() {
    $('#changeBtn').click(function() {
        changeBoxStyles();
    });
});

function changeBoxStyles() {
    // Change HTML making up the div:
    var name = $('#nameId').val();

    $('#box1').html("Hello <em>"
        + name + "</em>!");
}
```

Run when page is fully loaded.

Read contents of "name" input box.

Write HTML code into the div.

JQuery to Change Webpage

```
function changeBoxStyles() {

    console.log("Changing box styles.");


    var name = $('#nameId').val();
    $('#box1').html("Hello <em>" + name + "</em>!")


    $('#box2').html(
        '<h3>An Idea!</h3>'+
        '<p><img src="bell.png"
                alt="" width="80px"/></p>' +
        '<p>That\'s it!</p>');


    $('#box3').css({"border": "5px yellow",
                    "color":   "red",
                    "backgroundColor": "green"});


    $('#box4').hide();
}
```

Display browser console message

Read input field's text and use it.

Create complex html code from inside JavaScript code.

Style an element using CSS rules/properties

Hide the div (great for error displays)

! 10-StaticServer's: code.js

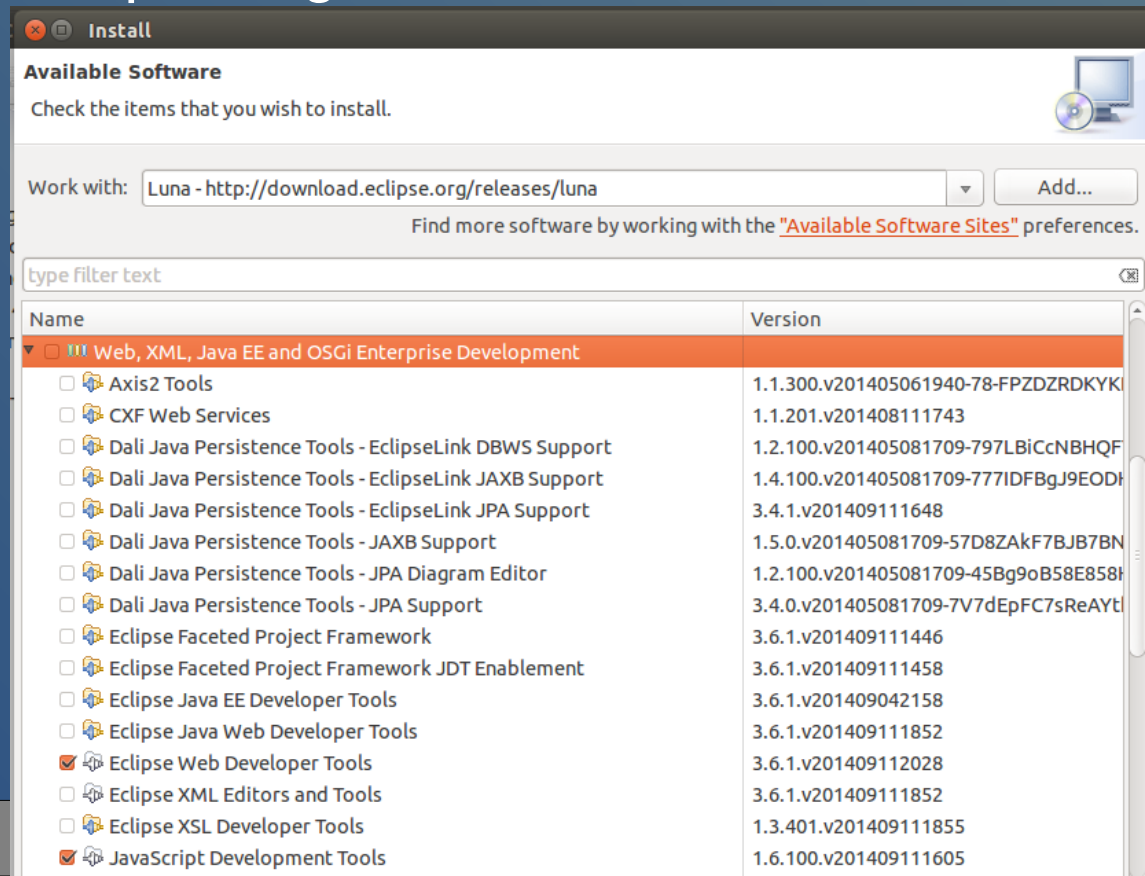
Client-Side Timers

```
$(document).ready(function() {
    window.setInterval(function() {updateTime()}, 1000);
});

function updateTime() {
    var now = new Date();
    var timeStr = now.getHours() + ':'
                + now.getMinutes() + ':'
                + now.getSeconds();
    $('#box3').html("Its now<br/>" + timeStr);
}
```

Call updateTime() every 1000ms (1sec)

Eclipse Setup

- Setup Eclipse to better handle .html, .css, .js files
  - Help --> Install New Software
  - Select update site for your version from drop-down, such as: “Mars – http://download.eclipse.org/releases/mars”
  - Under “Web, XML, Java EE...”
    - Eclipse Web Developer Tools
    - JavaScript Development Tools



Debugging Tools

- Browsers try to always make things work.
  - They usually quietly do their best to hide errors.
  - View error messages with the console (Firefox & Chrome F12)
  - Do this whenever you page is doing “funny” things.
- Validate your HTML to ensure it's correct.
  - Incorrect HTML can be rendered in unexpected ways.
  - https://validator.w3.org/

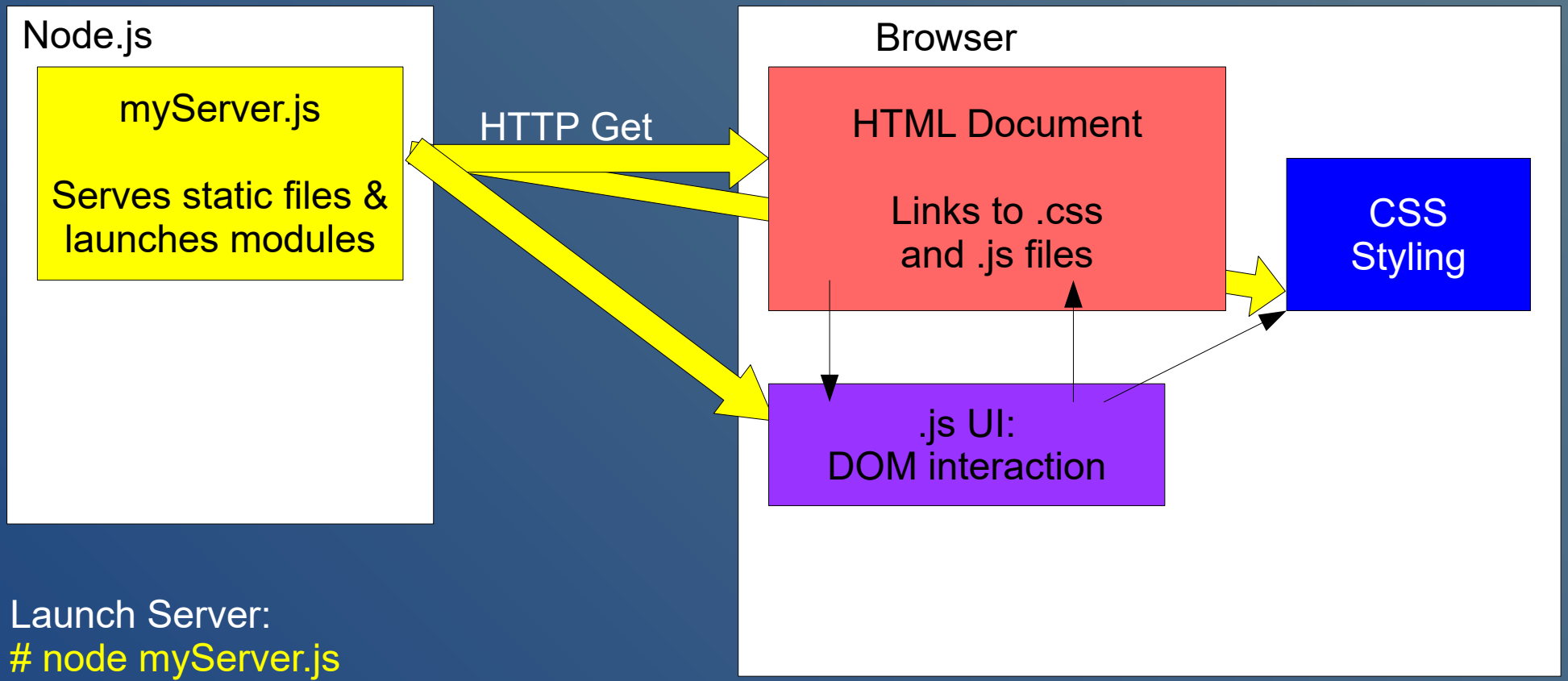
Serving Static Content with Node.js

Node.js is a platform built on Chrome's JavaScript runtime for building network applications.

Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

Serving Static Files to Client

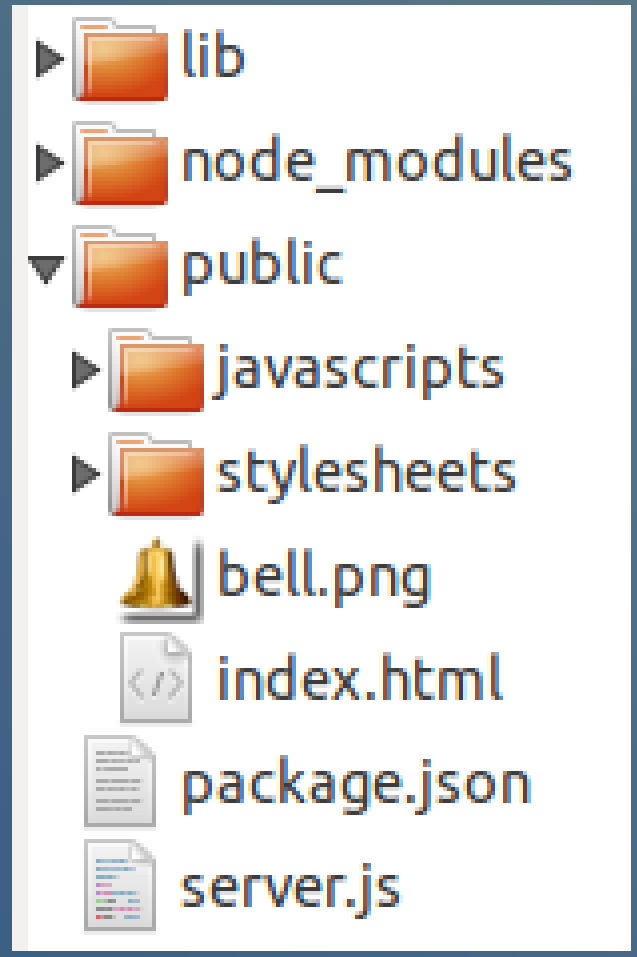
- Your Node.js server reads files from disk to send to client.



Launch Server:
node myServer.js

Suggested Server File Structure

- lib/
  Server side J.S. (more later...)
- node_modules/
  Modules installed by npm
- public/
  All client side static files
- /
  - package.json
    Configures server
  - server.js
    Server side starting logic.



Node.js Server

- Setup a Node.js application with a package.json file:

```
{
   "name": "demo-static-server",
   "version": "0.0.1",
   "description": "Demo Node.js server.",
   "dependencies": {
      "mime": "~1.2.7"
   }
}
```

- Install dependencies

  # npm install

- Run server

  # node myServer.js

Both work on host and target.

No need to cross-compile / recompile because..

JavaScript is interpreted / complied at runtime.

myServer.js (1/3)

```
var http = require('http');
var server = http.createServer(function(request, response) {
    var filePath = false;
    if (request.url == '/') {
        filePath = 'public/index.html';
    } else {
        filePath = 'public' + request.url;
    }
    var absPath = './' + filePath;
    serveStatic(response, absPath);
});

var PORT = 3042;
server.listen(PORT, function() {
    console.log("Server listening on port " + PORT);
});
```

..

Callback function created at startup, but..

executed later by an event.

Think of the event that triggers the function vs where the function is in the code.

Prints message to the server's terminal.

myServer.js (2/3)

```
var fs   = require('fs');
function serveStatic(response, absPath) {

    fs.exists(absPath, function(exists) {
        if (exists) {

            fs.readFile(absPath, function(err, data) {
                if (err) {
                    send404(response);
                } else {
                    sendFile(response, absPath, data);
                }
            });

        } else {
            send404(response);
        }
    });
}
```

Node.js is an asynchronous (non-blocking i/o) webserver:

All calls that could block.. use call-back functions.

myServer.js (3/3)

```
function send404(response) {
	response.writeHead(404, {'Content-Type': 'text/plain'});
	response.write('Error 404: resource not found.');
	response.end();
}

var mime = require('mime');
var path = require('path');
function sendFile(response, filePath, fileContents) {
	response.writeHead(
		200,
		{"content-type": mime.lookup(path.basename(filePath))}
	);
	response.end(fileContents);
}
```

Setup HTTP return packet:
Code (404)
Type (text/plain)
Content

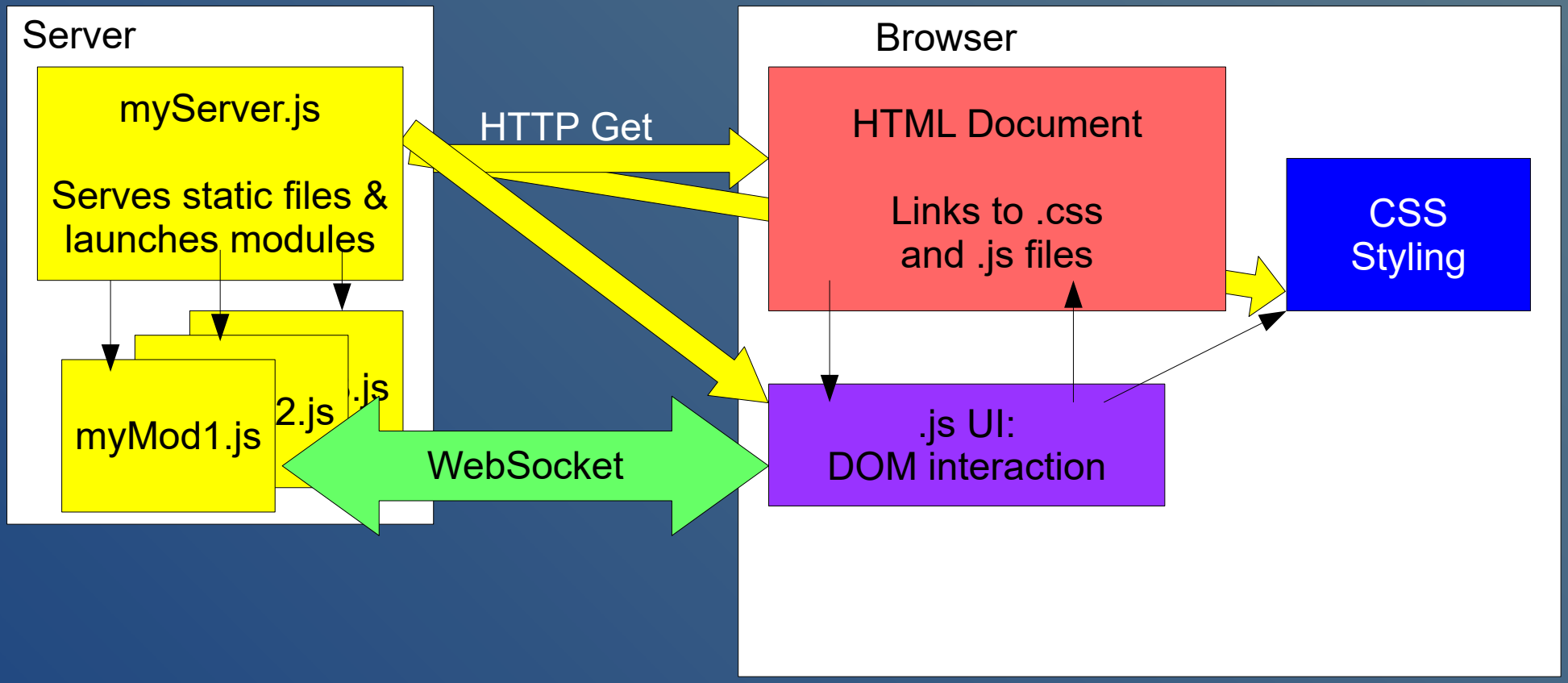
..

mime module figures out content type from file name.

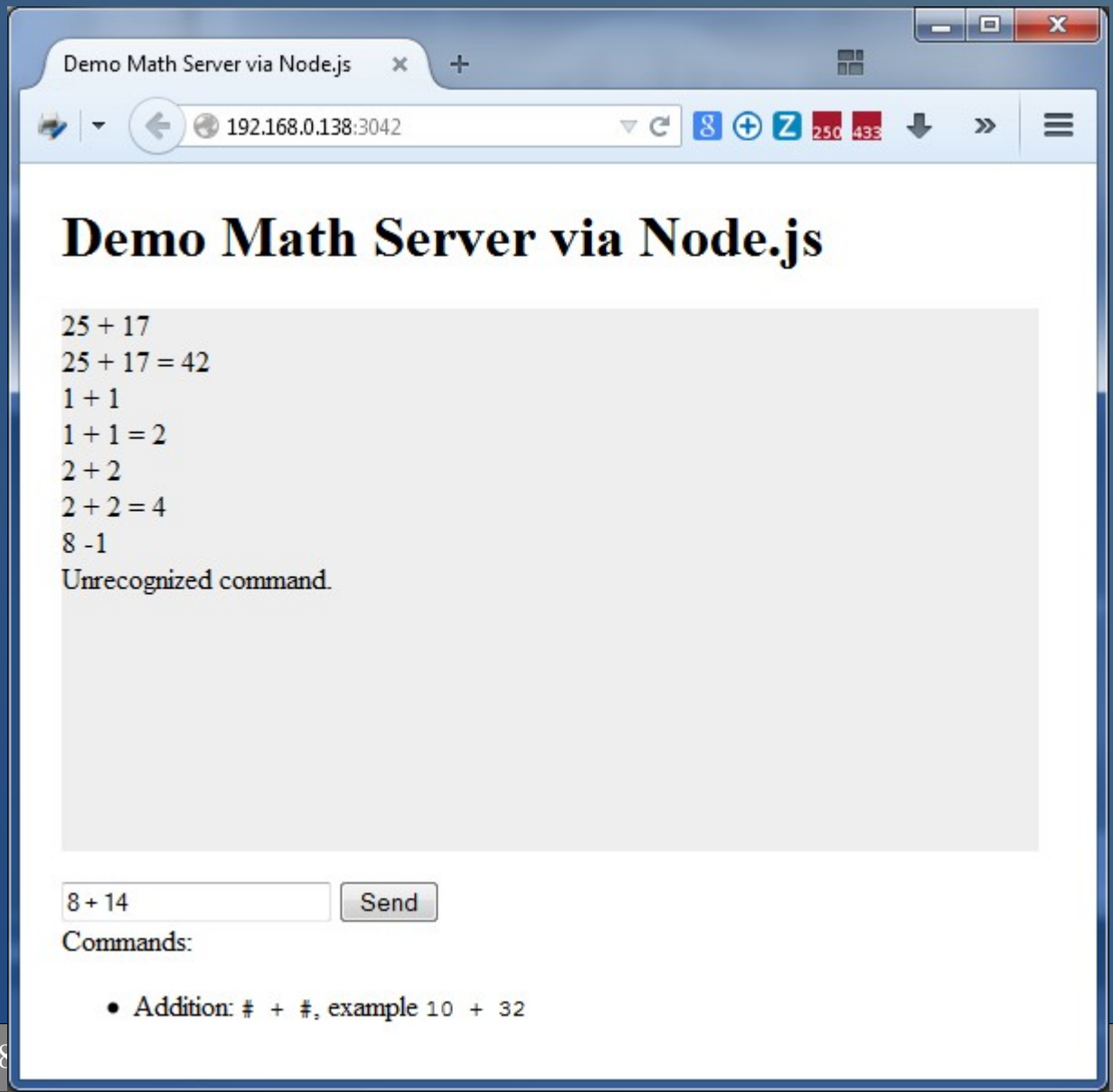
Dynamic Server Example with Node.js

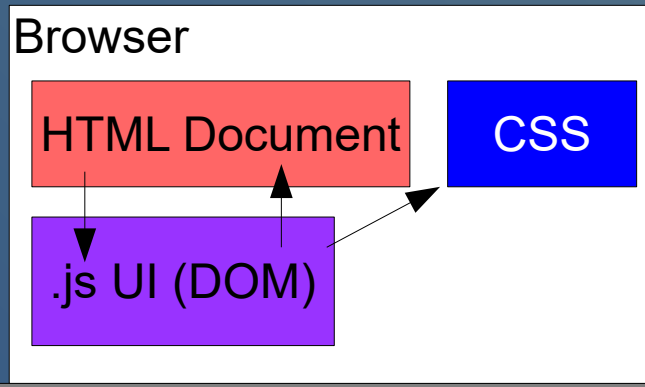
Dynamic Client Content

- WebSocket used to dynamically exchange messages.



Client: Webpage

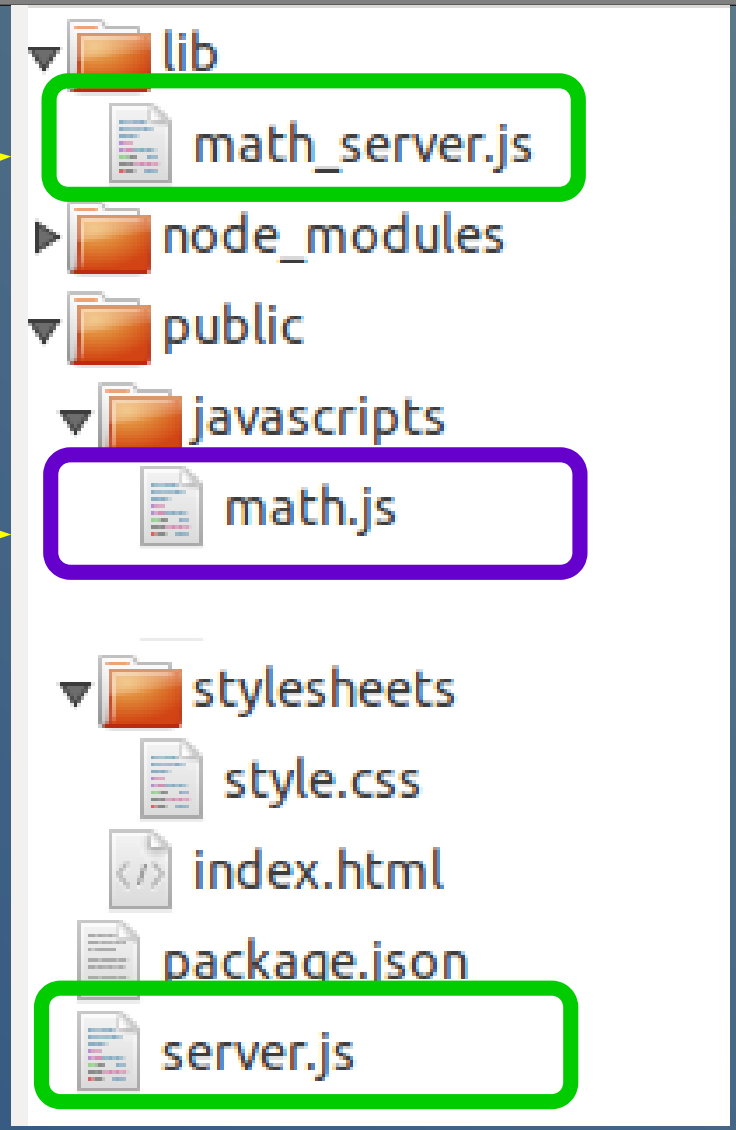




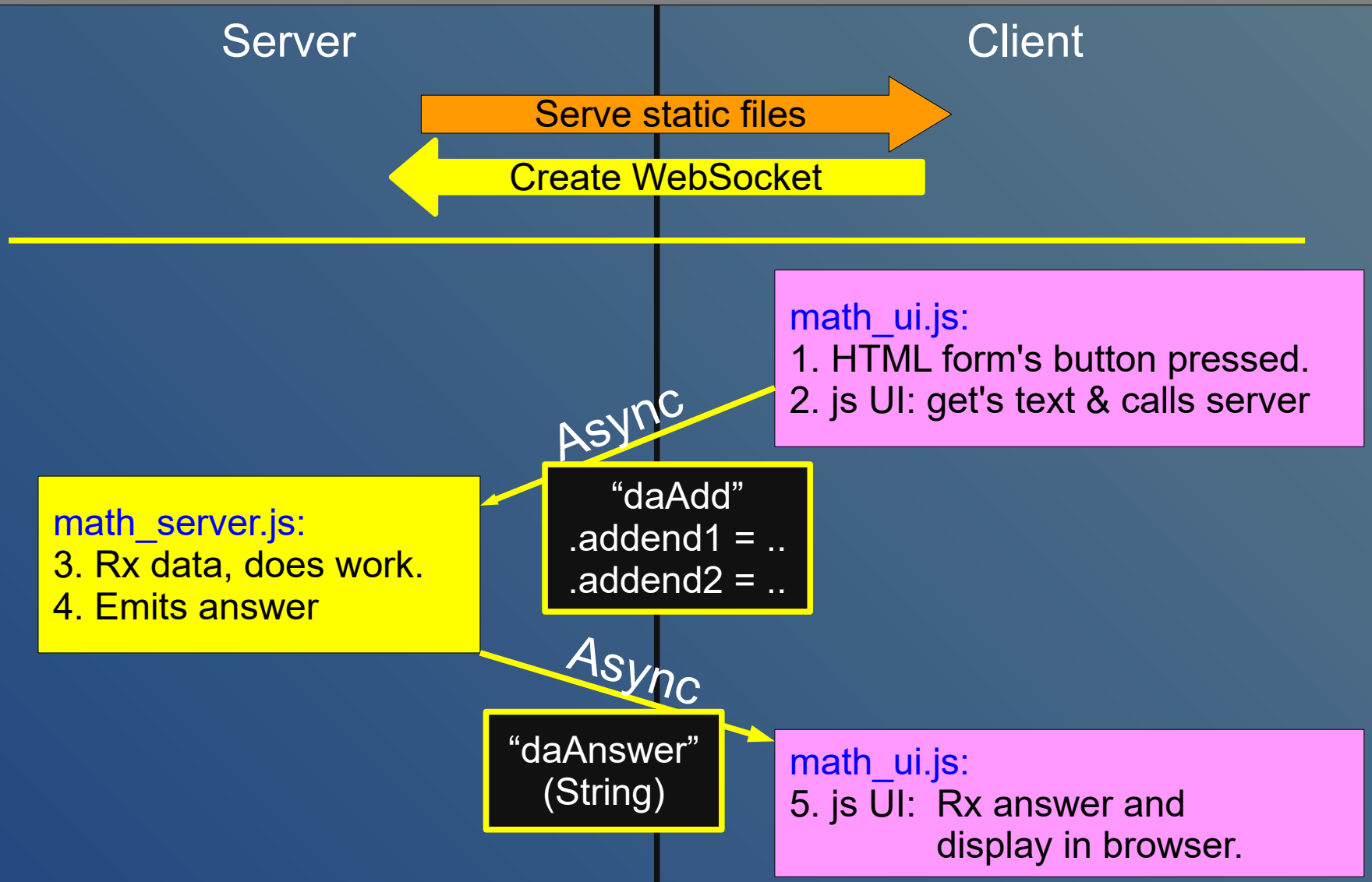
Suggested Server File Structure

Server Side Scripts

Client Side Script(s)



Dynamic Server: Sequence of calls



Dynamic Server Example

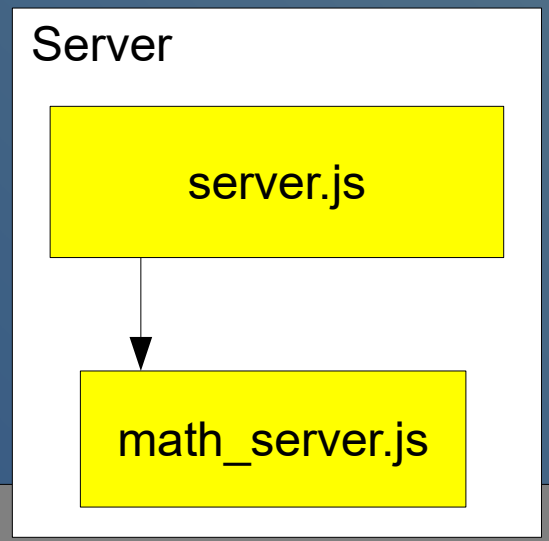
```
// Create the Math server to listen for the websocket
var mathServer = require('./lib/math_server');
mathServer.listen(server);
```

Add to end of server.js

File holds the static-content server, plus kicks-off our module.

```
var socketio = require('socket.io');
var io;

exports.listen = function(server) {
	io = socketio.listen(server);
	io.sockets.on('connection', function(socket) {
		handleCommand(socket);
	});
};

function handleCommand(socket) {
	// ... more on next slide.
};
```

Create custom module: ./lib/math_server.js



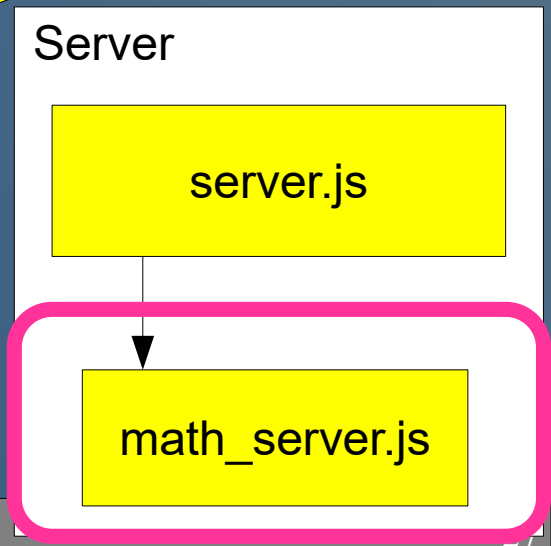
Dynamic Server cont. (math_server.js)

```
function handleCommand(socket) {
    socket.on('daAdd', function(data) {
        var val1 = Number(data.addend1);
        var val2 = Number(data.addend2)
        console.log('Adding ' + val1 + ' + ' + val2);

        var answer = doDaAddition(val1, val2);
        var message = val1 + ' + ' + val2 + ' = ' + answer;

        // Build and send reply.
        socket.emit('daAnswer', message);
    });
};
function doDaAddition(x, y) {
    return x + y;
}
```

Callback function for daAdd call.

Extract field from struct.

Send data over WebSocket

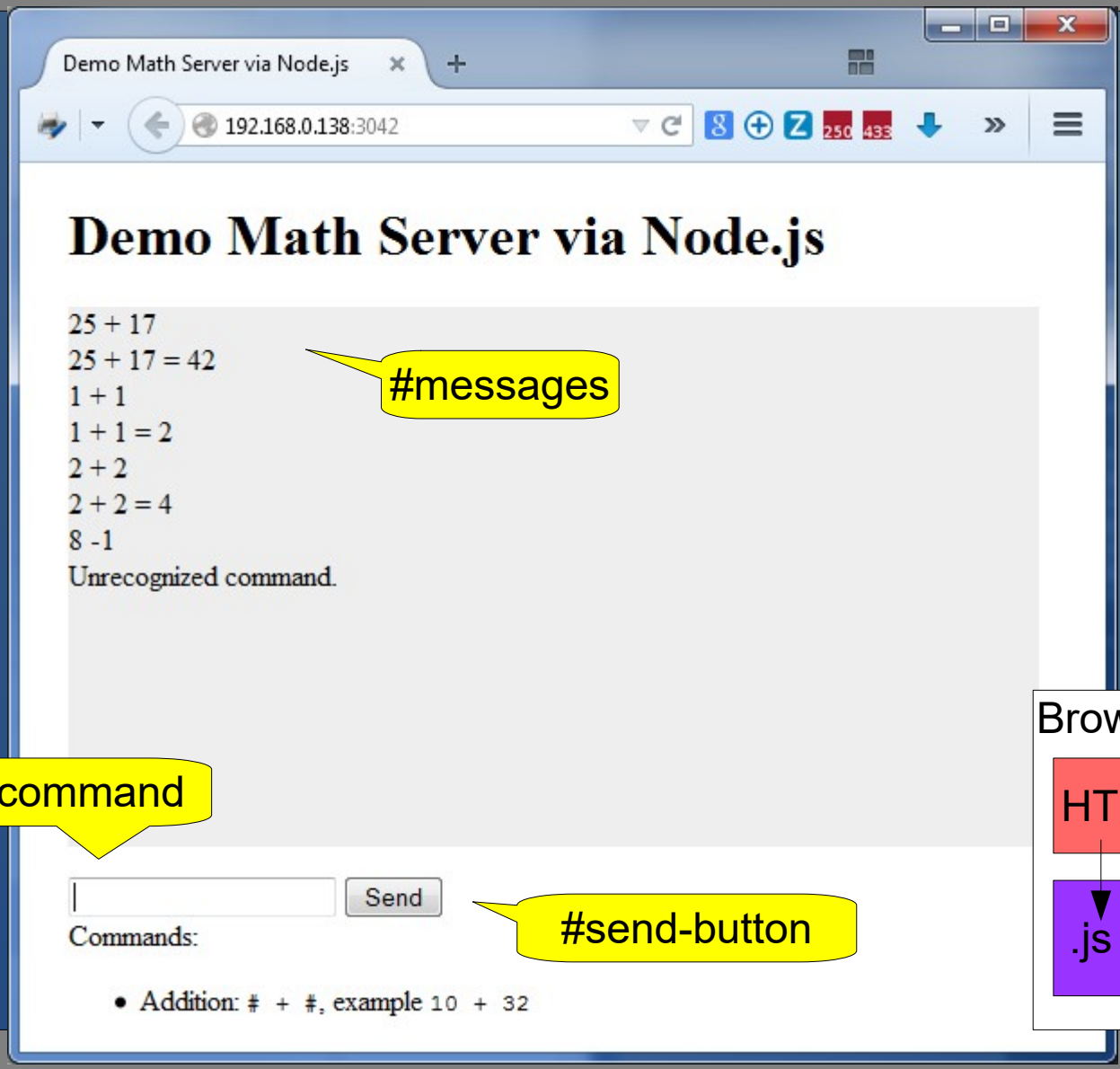


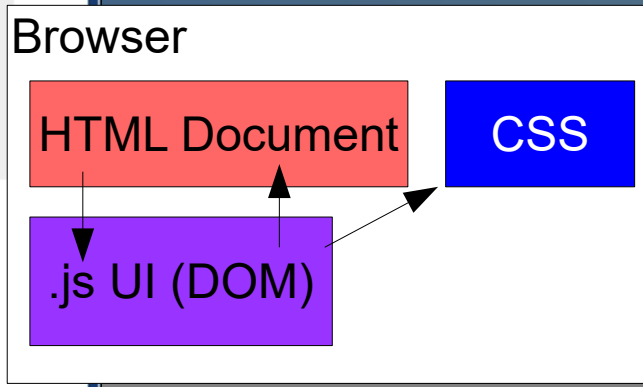
Server Timers

- Server-side timers are great for error timeouts.
  - Create a new timer and set what to run if it expires.
  - Elsewhere, clear timer when no longer needed.

```
function handleCommand(socket) {
   var errorTimer = setTimeout(function() {
      socket.emit("daError",
                  "Oops: Too slow!");
   }, 5000);

   socket.on('daAdd', function(data) {
      // ... code omitted...

      // Stop the timer:
      clearTimeout(errorTimer);
   });
};
```

Client: Webpage ID's





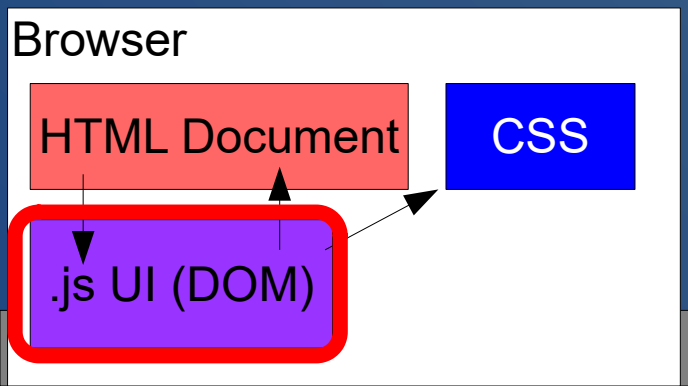
Client UI: Integrate with DOM (1/2)

./public/javascripts/math_ui.js

Execute function when page loaded

Callback for form's submit.

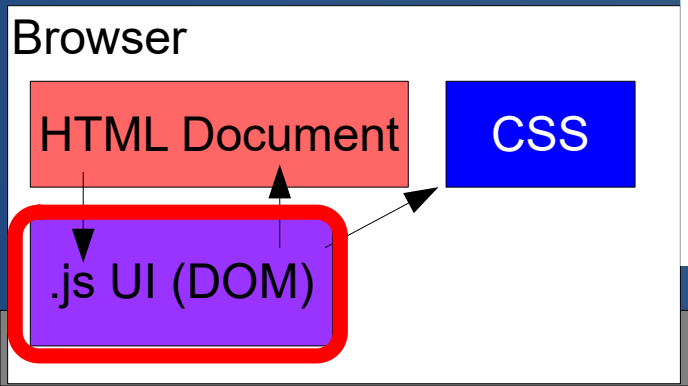
Create callback listening for “daAnswer” calls.




```
var socket = io.connect();
$(document).ready(function() {
    // Make the text-entry box have focus.
    $('#send-command').focus();

    // Allow sending the form.
    $('#send-form').submit(function() {
        readUserInput();

        // Return false to show we have handled it.
        return false;
    });

    // Handle data coming in from the server
    socket.on('daAnswer', function(result) {
        $('#messages').append(divMessage(result));
    });

});
```

Client UI: Integrate with DOM (2/2)

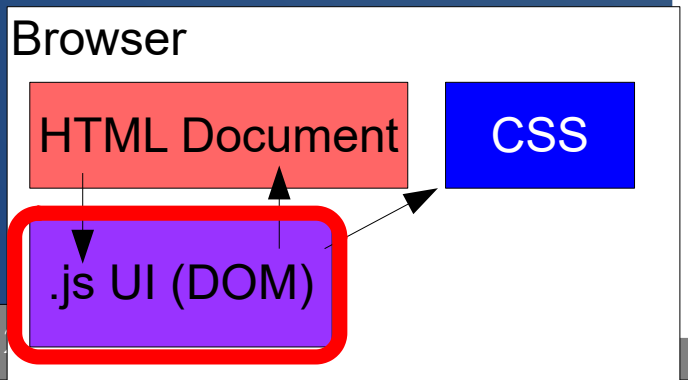
```
function readUserInput() {
    // Get the user's input from the browser.
    var message = $('#send-command').val();

    // Display the command in the message list.
    $('#messages').append(divMessage(message));

    // Process the command
    var errMsg = processCommand(message);
    if (errMsg) {
        $('#messages').append(divMessage(errMsg));
    }

    // Clear the user's command (ready for next command).
    $('#send-command').val('');
}

// Wrap a string in a new <div> tag
function divMessage(inString) {
    return $('<div></div>').text(inString);
}
```



Client UI: Interact with Server

Dynamically create a structure type.

"Emit" the message to the server. Give it a "message" name of 'daAdd'




```
function processCommand(command) {
    var words = command.split(' ');
    var operation = words[1];
    var message = false;

    switch(operation) {
    case '+':
        var request = {
                addend1: Number(words[0]);,
                addend2: Number(words[2]);
        };
        socket.emit('daAdd', request);
        break;
    default:
        message = 'Unrecognized command.'
    }
    return message;
};
```

Node.js to C App (UDP)

Text in Webpage

- JavaScript can insert text/content into the web page.
- HTML: Use and
  - For dynamic content.. in its own paragraph:

  - For dynamic content.. inside a paragraph:

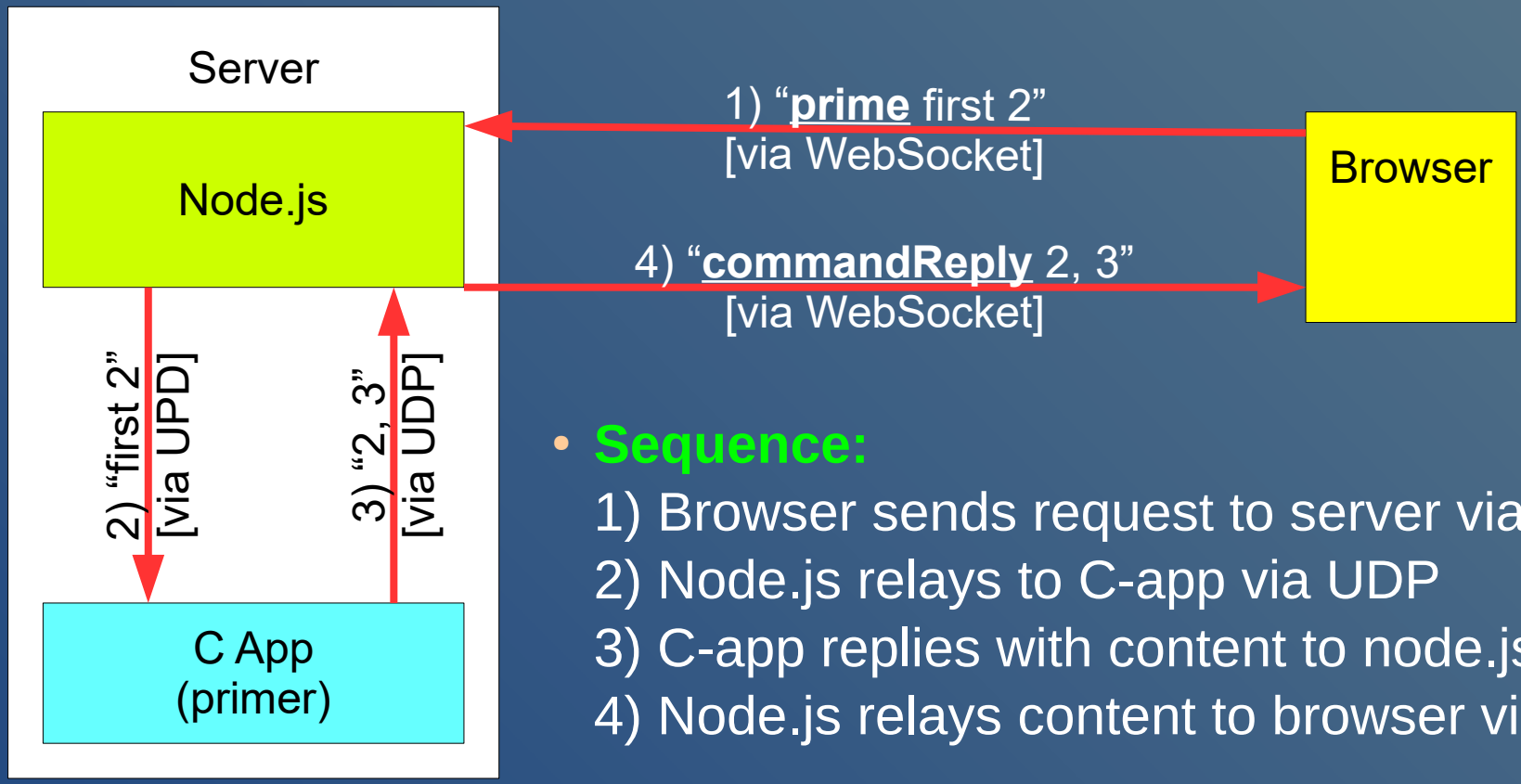
- JavaScript
  - $('#daName').html('My dynamic content');

Reading Files

- Node.js on the server reads files using “fs” module
  - Used in our “static” page server.
  - Can also be used for reading /proc files
- Details
  - File data comes back as a character array.
    Convert to a string:
    var str = daFileContents.toString('utf8');
  - Possible security problem:
    Allowing client to request a file: may have ../ in path.

Node.js and C Sockets

- Use UDP socket for Node.js server to communicate with a local C/C++ application.



- **Sequence:**
  1) Browser sends request to server via websocket
  2) Node.js relays to C-app via UDP
  3) C-app replies with content to node.js via UDP
  4) Node.js relays content to browser via websocket

FYI: HTTPS

- Use HTTPS for secure, non-sniffable communication

1. Generate private key in base folder of project
   $ openssl genrsa 1024 > key.pem
2. Generate public certificate (unsigned)
   $ openssl req -x509 -new -key key.pem > key-cert.pem
3. Code changes from non-HTTPS:
   a) require('https')
   b) options struct for private/public key
   c) pass options to http.createServer
4. HTML: Use https:// (vs http://) to link to jQuery:

```
<script src='https://code.jquery.com/jquery-1.8.0.min.js'
        type='text/javascript'></script>
```

Summary

Node.js Troubleshooting:
Error: No such file or directory
at Function.resolveArgv0 (node.js:289:23)
at startup (node.js:43:13)
at node.js:448:3
Run the following on your BeagleBone in the server's folder:
sudo npm cache clean -f
sudo npm install -g n

- Client Side:
  - .html for static page content
  - .css for look
  - UI .js for DOM interaction & WebSocket
- Server Side:
  - Serve static pages
  - Module(s) for dynamic content via WebSocket
- Node.js: JavaScript based web-server platform.
  - and to insert text into web page.
  - “fs” module to read from /proc/ files (or others).
  - UDP socket to access C/C++ application.