To use the HTTP server and client

The path module provides utilities for working with file and directory paths

Async provides around 20 functions that include the usual 'functional' suspects (map, reduce, filter, each…) as well as some common patterns for asynchronous control flow (parallel, series, waterfall…). All these functions assume you follow the Node.js convention of providing a single callback as the last argument of your async function.

Socket.IO enables real-time bidirectional event-based communication. It works on every platform, browser or device, focusing equally on reliability and speed. Socket.IO is built on top of the WebSockets API (Client side) and Node.js. It is one of the most depended upon library on **npm** (Node Package Manager).

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework −

- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).

When our server is created, it registers a request handler to which the server will delegate the request handling

It prepares itself to receive requests via socket.io. Here, we are expecting to receive just one type of message: the connection.

use() method mounts the middleware express.static for every request. The express.static **middleware** is responsible for serving the static assets of an Express.js application. The express.static() method specifies the folder from which to serve all static resources.

Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application’s request-response cycle. The next middleware function is commonly denoted by a variable named next.

Middleware functions can perform the following tasks:

- Execute any code.
- Make changes to the request and the response objects.
- End the request-response cycle.
- Call the next middleware function in the stack.

This will normalise all the arguments into a path string. This can come in really handy using the **__dirname** global and a folder / file. This will set your view folder to something like:/Project/whatever/views

Explicitly - If you don't set view engine in Express.js then the extension of template must be specified explicitly to res.render() method e.g. res.render('view.jade', data);

On - Start listening for server-sent events from with the specified event handler. Will trigger the provided callback function when a matching event is received from the client-side.

broadcasting means sending a message to all connected clients. Broadcasting can be done at multiple levels. We can send the message to all the connected clients, to clients on a namespace and clients in a particular room. To broadcast an event to all the clients, we can use the **io.sockets.emit** method.

The get and set functions on the socket object were removed in version 1.x. The proper way to store and retrieve values now is through properties on the socket object.

Eg
```
socket.on('set nickname', function (name) {
        socket.nickname = name;
});
```
https://github.com/socketio/socket.io/blob/master/examples/chat/index.js#L35-L36

How to update: npm i socket.io@latest --save

Send a socket request (virtual GET) to server,

if we want to send an event to everyone, but the client that caused it (previously - it was caused by new clients on connecting), we can use the **broadcast**

Io.connect - specify ip address of client to connect to the server https://docs.angularjs.org/guide/

https://www.w3schools.com/angular/default.asp

AngularJS is a very powerful JavaScript Framework. It is used in Single Page Application (SPA) projects. It extends HTML DOM with additional attributes and makes it more responsive to user actions. AngularJS is open source,https://docs.angularjs.org/guide/component

Angular 2 is the current version. -- https://angular.io/guide/upgrade

AngularJS is a **JavaScript framework**, extends HTML attributes with **Directives**, and **binds** data to HTML with **Expressions**, and starts automatically when the web page has loaded.

Directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler ($compile) to attach a specified behavior to that DOM element (e.g. via event listeners), or even to transform the DOM element and its children.

Expressions are like JavaScript expressions with the following differences:

- **Context**: JavaScript expressions are evaluated against the global window. In AngularJS, expressions are evaluated against a scope object.
- **Forgiving**: In JavaScript, trying to evaluate undefined properties generates ReferenceError or TypeError. In AngularJS, expression evaluation is forgiving to undefined and null.
- **Filters**: You can use filters within expressions to format data before displaying it.
- **No Control Flow Statements:** You cannot use the following in an AngularJS expression: conditionals, loops, or exceptions.
- **No Function Declarations:** You cannot declare functions in an AngularJS expression, even inside ng-init directive.
- **No RegExp Creation With Literal Notation**: You cannot create regular expressions in an AngularJS expression. An exception to this rule is ng-pattern which accepts valid RegExp.
- **No Object Creation With New Operator:** You cannot use new operator in an AngularJS expression.
- **No Bitwise, Comma, And Void Operators:** You cannot use Bitwise, , or void operators in an AngularJS expression.

Controllers - a JavaScript constructor function that is used to augment the AngularJS Scope. When a Controller is attached to the DOM via the ng-controller directive, AngularJS will instantiate a new Controller object, using the specified Controller's constructor function. A new child scope will be created and made available as an injectable parameter to the Controller's constructor function as **$scope.**

The **scope** is the binding part between the HTML (view) and the JavaScript (controller).

If the controller has been attached using the controller as syntax then the controller instance will be assigned to a property on the new scope.

Module - a container for the different parts of your app – controllers, services, filters, directives, etc.   eg ng-app

These are directives: https://www.w3schools.com/angular/angular_ref_directives.asp

Ng-bind: transform DOM element like document.getElementByID(“users”).innerHTML

Ng-repeat: for each loop

Ng-model: binds the value of HTML controls (input, select, textarea) to application data.

Ng-change:

- tells AngularJS what to do when the value of an HTML element changes.
- requires a ng-model directive to be present.
- from AngularJS will not override the element's original onchange event, both the ng-change expression and the original onchange event will be executed.
- event is triggered at every change in the value. It will not wait until all changes are made, or when the input field loses focus.
- event is only triggered if there is a actual change in the input value, and not if the change was made from a JavaScript.

Ng-disabled: sets the disabled attribute of a form field (input, select, or textarea).

https://docs.angularjs.org/guide/unit-testing