EXPRESS & EJS

COMPS381F

REFERENCES

- https://scotch.io/tutorials/use-ejs-to-template-your-node-application

SERVER-SIDE MVC

MVC: MODEL, VIEW, CONTROLLER

- Separate the user interface (the **View**) from the underlying data (the **Model**). Logic (usually business logic) in the **Controller** is used to bind the model and the view
- This separation of concerns helps to organize code, making it quicker to developer, less expensive to maintain, and easier to test.

MVC: BASIC CONCEPTS

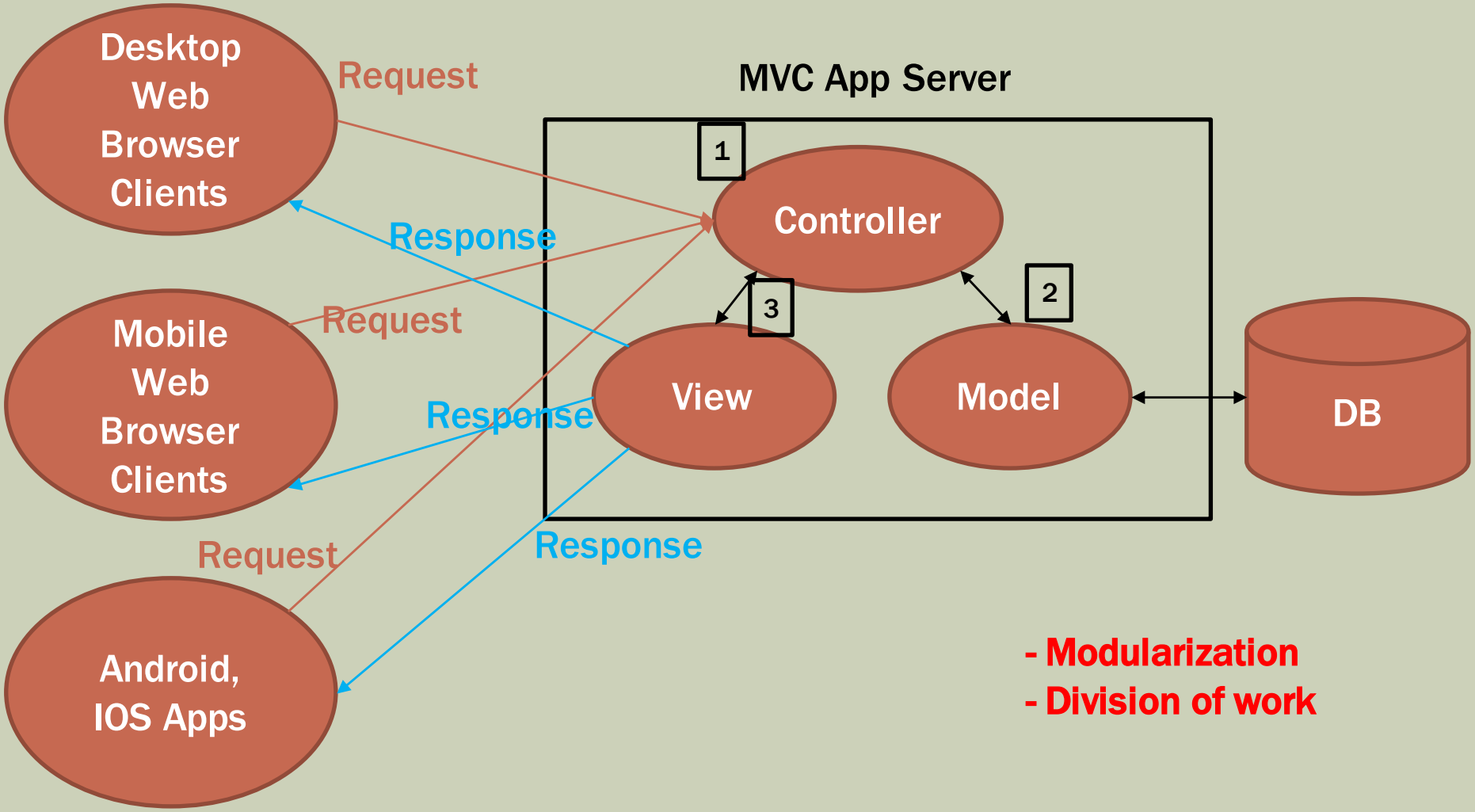
Mongo ose (ODM)

- **Model**
  - The model represents the data used by an application, and the rules to manipulate that data. This includes **loading data from a database**, and **validating data** before storing it.
- **View**
  - The view represents the **user interface** of an application. A view is rendered by the application in **response to a request**, at which point it is displayed to the user in a web browser (or mobile apps).

EJS

- **Controller**
  - The controller provides the **glue between the model and the view**. Controllers process incoming requests from the web browser, loading data from models and passing that data on to a view for presentation.

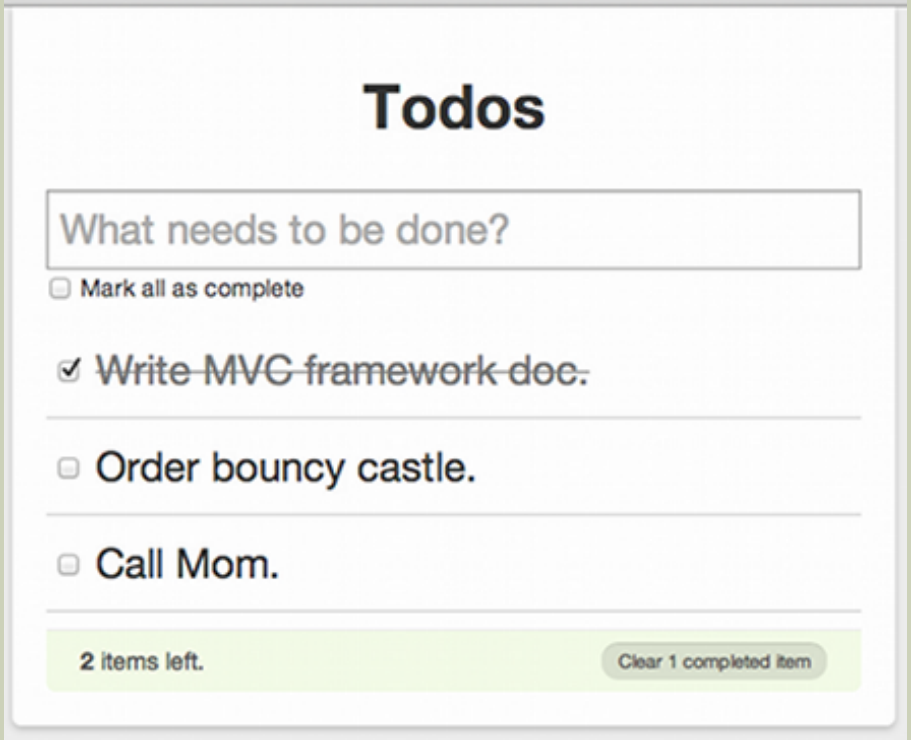
MVC - OVERVIEW



- Modularization
- Division of work

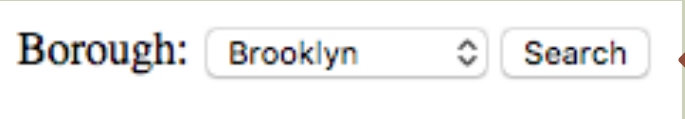
MVC – EXAMPLE (1)

- **Model** represents attributes associated with each to-do item such as description and status
  - When a new to-do item is created, it is stored in an instance of the model.
- **View** is what's presented to the users and how users interact with the app.
  - The view is usually made with HTML, CSS, JavaScript and often templates.
- **Controller** is the decision maker
  - The glue between the model and view
  - The controller updates the view when the model changes.
  - It also adds event listeners to the view and updates the model when the user manipulates the view.



MVC EXAMPLE (2)

View



Model

```
findRestaurants(),
findDistinctBorough()
```

Controller

```
http.createServer(function (req,res) {
…
if (parsedURL.pathname == '/show') …
if (parsedURL.pathname == "/") …
```

```
if (parsedURL.pathname == '/show') {
        console.log(queryAsObject);
        MongoClient.connect(mongodbURL, function(err, db) {
            assert.equal(null, err);
            findRestaurants(db, queryAsObject, function() {
                res.writeHead(200, {"Content-Type": "application/json"});
                res.write(JSON.stringify(restaurants));
                res.end();
                db.close();
                console.log(today.toTimeString() + " " + "CLOSED CONNECTION "
                            + req.connection.remoteAddress);
            });
        });
    }
    else if (parsedURL.pathname == "/") { // display HTML form
        MongoClient.connect(mongodbURL, function(err, db) {
            assert.equal(null, err);
            findDistinctBorough(db, function() {
                res.writeHead(200, {"Content-Type": "text/html"});
                //res.write(JSON.stringify(boroughs));
                res.write("<html><body>");
                res.write("<form action=\"/show\" method=\"get\">");
                res.write("Borough: ");
                res.write("<select name=\"borough\">");
                for (i in boroughs) {
                    res.write("<option value=\"" +
                        boroughs[i] + "\">" + boroughs[i] + "</option>");
                }
                res.write("</select>");
                res.write("<input type=\"submit\" value=\"Search\">");
                res.write("</form>");
                res.write("</body></html>");
                res.end();
                db.close();
                console.log(today.toTimeString() + " " + "CLOSED CONNECTION "
                            + req.connection.remoteAddress);
            });
        });
```

MVC EXAMPLE (3)

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');

var kittySchema = require('./models/kitty');
var db = mongoose.connection;

db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function (callback) {
    var Kitten = mongoose.model('Kitten', kittySchema);
    var fluffy = new Kitten({name: 'fluffy', age: 0});

    fluffy.save(function(err) {
        if (err) throw err
        console.log('Kitten created!')
        db.close();
    });
});
```

Model

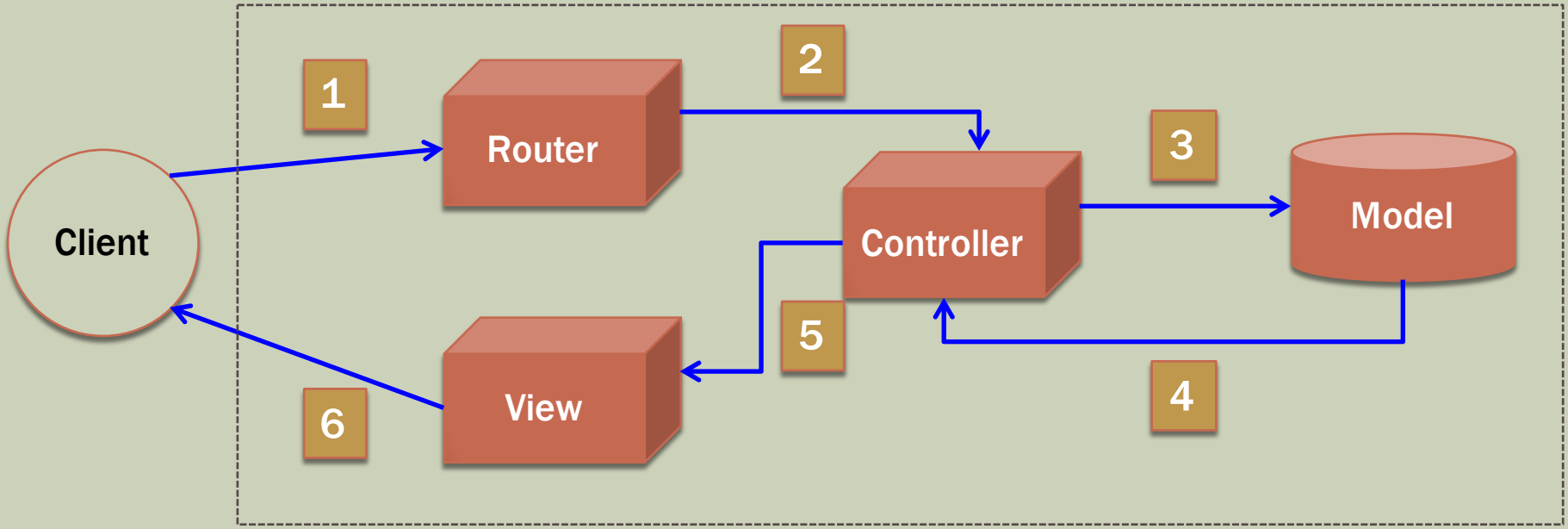
DESIGN FOR MVC

- Consider the following requirements:
  - A NodeJS server, upon receiving GET requests, displays all documents in the `Kittens` collection
  - Enforce the following *business rule*:
    - If the client request is initiated by an "admin" user, the server displays both the `name` and `age` fields; only the `name` field otherwise.

HIGH-LEVEL DESIGN

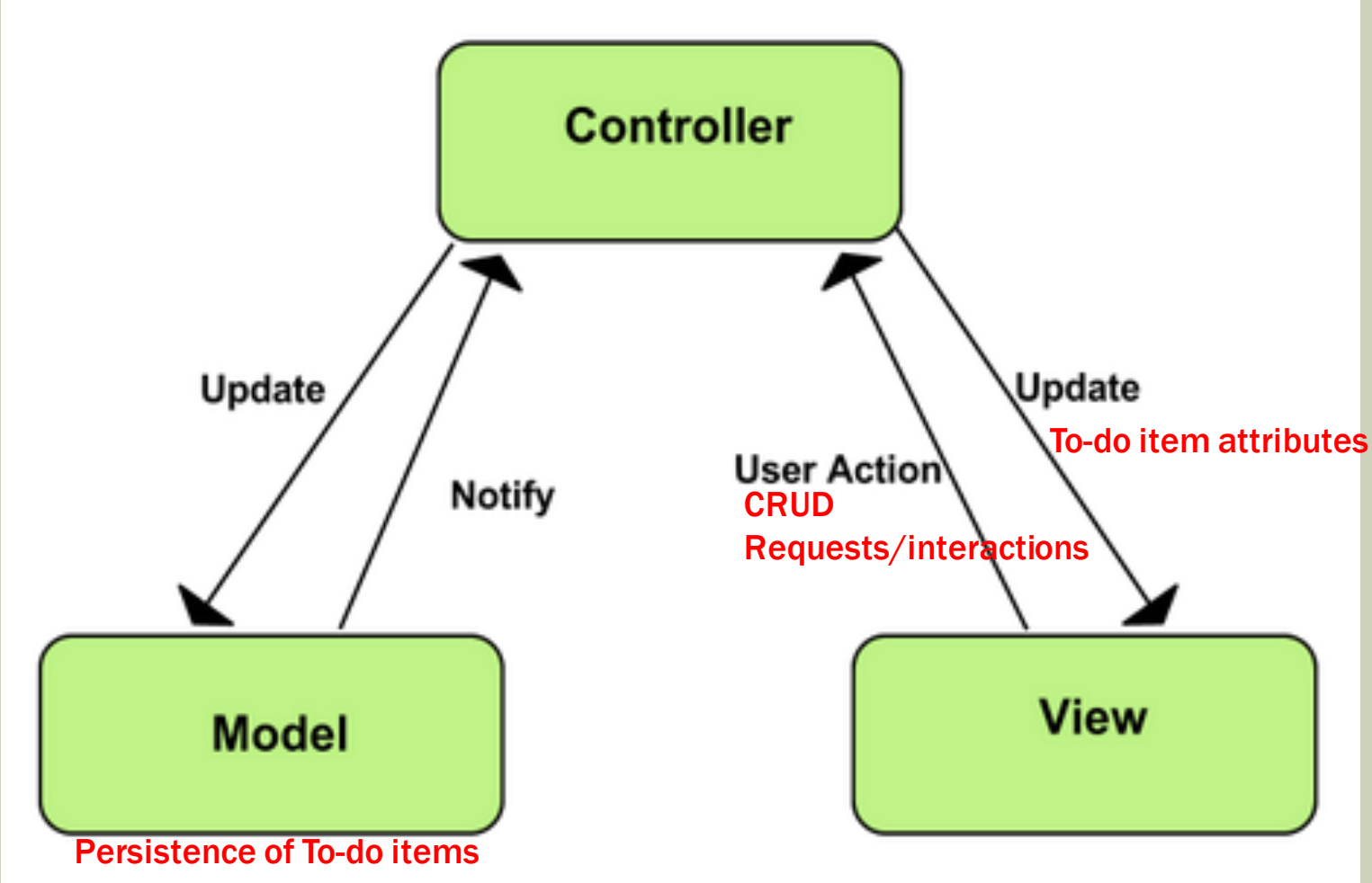
- GET Requests:
  - Admin users: `/show?id=admin`
    - `Pathname = /show`
    - `Query String: id=admin`
  - Other users: `/show`
    - `Pathname = /show`
    - `Query String: Nil`
- View
  - `renderResults()` – display documents
- Controller
  - `filterResult()` – control fields to be displayed
    - Admin users: name and age fields
    - Other users: name field only
- Source Code
  - https://github.com/raymondwcs/mongoose/blob/master/mvcfind/server.js

MVC APP ARCHITECTURE



1. A request comes into your application
2. (Optional) The request gets routed to a controller
3. The controller, if necessary, makes requests to the model
4. The model responds to the controller
5. The controller sends a response to a view
6. The view *renders* a response to the original requester

MVC DESIGN PATTERN (TO-DO EXAMPLE)



ADVANTAGES OF MVC

- Reusable and extendable code.
- Separation of view logic from business logic.
- Allow simultaneous work between developers who are responsible for different components (such as UI layer and core logic).
- Easier to maintain.

EXPRESS

WHAT IS EXPRESS?

- Express (http://expressjs.com) is a lightweight **framework** for Node.js applications
  - Uses a disciplined directory structure for organizing source code files. For instance:

```
appname/
appname/package.json
appname/server.js
appname/models
appname/views
```

  - Supports the concepts of MVC
    - Allows the **use of template engine** (EJS, for instance) to inject data into HTML code
    - **App routing** and session control

A BASIC EXPRESS APP (1)

```
{
    "name": "server",
    "description": "NodeJS Express App",
    "author": "Raymond SO <rso@ouhk.edu.hk>",
    "version": "0.0.1",
    "dependencies": {
        "mongoose": "*",
        "express" : "*"
    },
    "engine": "node >= 0.10.0",
    "scripts": {"start": "node server"}
}
```

A BASIC EXPRESS APP (2)

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send(greetingMsg(req.query.name,false));
});

app.get('/greetings', function (req, res) {
  res.send(greetingMsg(req.query.name,false));
});

app.get('/greetings/sayHello', function (req, res) {
  res.send(greetingMsg(req.query.name,false));
});

app.get('/greetings/sayHelloWithTime', function (req, res) {
  res.send(greetingMsg(req.query.name,true));
});

var server = app.listen(8099, function () {
  var host = server.address().address;
  var port = server.address().port;

  console.log('Server listening at http://%s:%s', host, port);
});
```

```
function greetingMsg(name,showtime) {
    var today = new Date();
    var msg = "";
    if (name != null) {
        msg = "Hello " + name + "! ";
    }
    else {
        msg = "Hello! ";
    }
    if (showtime) {
        msg += " It is now " + today.toTimeString();
    }
    return(msg);
}
```

App routing example

Source: https://github.com/raymondwcs/express/blob/master/greetings/server.js

BASIC ROUTING

```
// respond with "Hello World!" on the homepage
app.get('/', function (req, res) {
  res.send('Hello World!');
});

// accept POST request on the homepage
app.post('/', function (req, res) {
  res.send('Got a POST request');
});

// accept PUT request at /user
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user');
});

// accept DELETE request at /user
app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user');
});
```

More details about routing can be found: http://expressjs.com/guide/routing.html

SERVING STATIC FILES

- Specify the directory that is to be marked as the location of static assets (such as images, CSS, JavaScript and etc.)

```
app.use(express.static('public'));
```

- Load the static files under the public directory such as:

http://localhost:3000/images/kitten.jpg
http://localhost:3000/css/style.css
http://localhost:3000/js/app.js
http://localhost:3000/images/bg.png
http://localhost:3000/hello.html

USING MULTIPLE DIRECTORIES FOR STATIC FILES

- To use multiple directories for serving static files, call the `express.static` function multiple times:

```
app.use(express.static('public'));
app.use(express.static('files'));
```

- The files will be looked up in the order the static directories were set
- Alternatively, use "virtual" paths:

```
app.use('/static', express.static('public'));
```

  - Load the files under the public directory, from the path prefix "/static" such as:
    http://localhost:3000/static/images/kitten.jpg
    http://localhost:3000/static/css/style.css
    http://localhost:3000/static/js/app.js
    http://localhost:3000/static/images/bg.png
    http://localhost:3000/static/hello.html

EJS

REFERENCES

- http://www.embeddedjs.com
- https://codeforgeek.com/2014/06/express-nodejs-tutorial/
- https://codeforgeek.com/2014/10/express-complete-tutorial-part-2/

ABOUT EJS

- EJS – Embedded JavaScript
- EJS combines *data* and a *template* to produce HTML

Instead of:

```
var html = "<h1>"+data.title+"</h1>"
html += "<ul>"
for(var i=0; i<data.supplies.length; i++) {
    html += "<li><a href='supplies/"+data.supplies[i]+"'>"
    html += data.supplies[i]+"</a></li>"
}
html += "</ul>"
```

Separate view(s) from programming (business) logic:

HTML = Template + Data

Cleaning Supplies
* mop
* broom
* duster

```
<h1><%= title %></h1>
<ul>
<% for(var i=0; i<supplie
   <li><%= supplies[i] %>
<% } %>
</ul>
```

```
{ title:
    'Cleaning Supplies',
  supplies:
    ['mop',
     'broom',
     'duster'] }
```

```
html= new EJS({url:'template.ejs'}).render(data)
```

INJECTING DATA TO VIEW

server.js

```
app.get('/',function(req,res){

        res.render('index',{title:"Home page"});

});
```

Send value of the title variable to your *view*

views/index.ejs

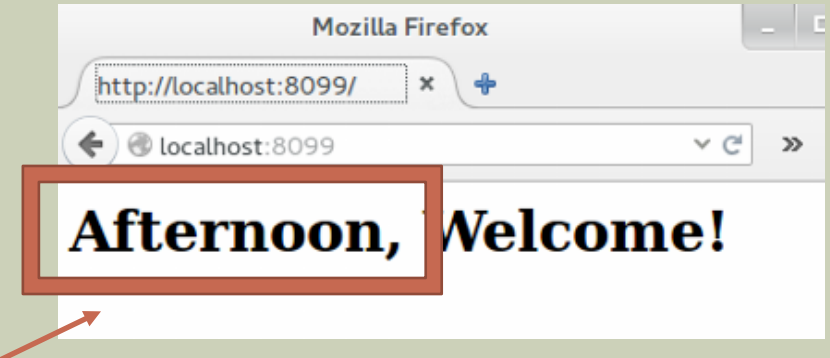
```
<html>
    <head>
        <title><%= title %></title>
    </head>
</html>
```

Access *title* variable in your view

SIMPLE EJS EXAMPLE

- Let's say we need to render a HTML page with a greeting message that depends on the time of the day.

```
function greetMessage() {
    var dateToday = new Date();
    var theHour = dateToday.getHours();
    if (theHour > 18) {
        gMessage = 'Evening';
    } else if (theHour > 12) {
        gMessage = 'Afternoon';
    } else {
        gMessage = 'Morning';
    }
    return gMessage;
}
```



GOOGLE MAP – EJS EXAMPLE

```
<html>
    <head>
        <title>Google Map Example</title>
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=true"></script>
        <script type="text/javascript">
            var loadMap = function()
            {
                var myLatLng = {lat: 22.316109, lng: 114.180459};
                var map = new google.maps.Map(document.getElementById("map"),{
                      zoom: 18,
                      center: myLatLng
                });
                var map = new google.maps.Marker({
                    position: myLatLng,
                    map: map,
                    title: 'OUHK'
                });
            };
            window.onload= loadMap;
        </script>
    </head>
    <body>
        <div id="map" style="width:500px;height:500px;"></div>
    </body>
</html>
```

GOOGLE MAP EJS EXAMPLE

views/gmap.ejs

```
<html>
    <head>
        <title>Google Map Example</title>
        <script type="text/javascript"
                 src="http://maps.google.com/maps/api/js?sensor=true"></script>
        <script type="text/javascript">
            var loadMap = function()
            {
                var myLatLng = {lat: <%=lat%>, lng: <%=lon%>};
                var map = new google.maps.Map(document.getElementById("map"),{
                      zoom: <%=zoom%>,
                      center: myLatLng
                });
                var map = new google.maps.Marker({
                    position: myLatLng,
                    map: map,
                    title: 'OUHK'
                });
            };
            window.onload= loadMap;
        </script>
    </head>
    <body>
        <div id="map" style="width:500px;height:500px;"></div>
    </body>
</html>
```

Source: https://github.com/raymondwcs/ejs/blob/master/googlemap/views/gmap.ejs

GOOGLE MAP EJS EXAMPLE

server.js

```
var express = require('express');
var app = express();

app.use(express.static(__dirname + '/graphics'));

app.get("/", function(req,res) {
    var lat  = req.query.lat;
    var lon  = req.query.lon;
    var zoom = req.query.zoom;

    res.render("gmap.ejs",{lat:lat,lon:lon,zoom:zoom});
});

app.listen(process.env.PORT || 8099);
```

http://localhost:8099/?lat=22.316109&lon=114.180459&zoom=18

Source: https://github.com/raymondwcs/ejs/blob/master/googlemap/server.js

INSTALL DEPENDENCIES

```
{
    "name": "server",
    "description": "NodeJS EXPRESS EJS app",
    "author": "Raymond SO <rso@ouhk.edu.hk>",
    "version": "0.0.1",
    "dependencies": {
        "mongoose": "*",
        "express": "*",
        "ejs": "*"
    },
    "engine": "node >= 0.10.0",
    "scripts": {"start": "node server"}
}
```

CREATE AN EJS TEMPLATE

- Create a template file: `views/welcome.ejs`

```
<HTML><BODY>
<H1><%= greetMsg %>, Welcome!</H1>
</BODY></HTML>
```

Varaible sent by the express nodejs server (the controller)

TELL YOUR NODEJS APP ABOUT EJS

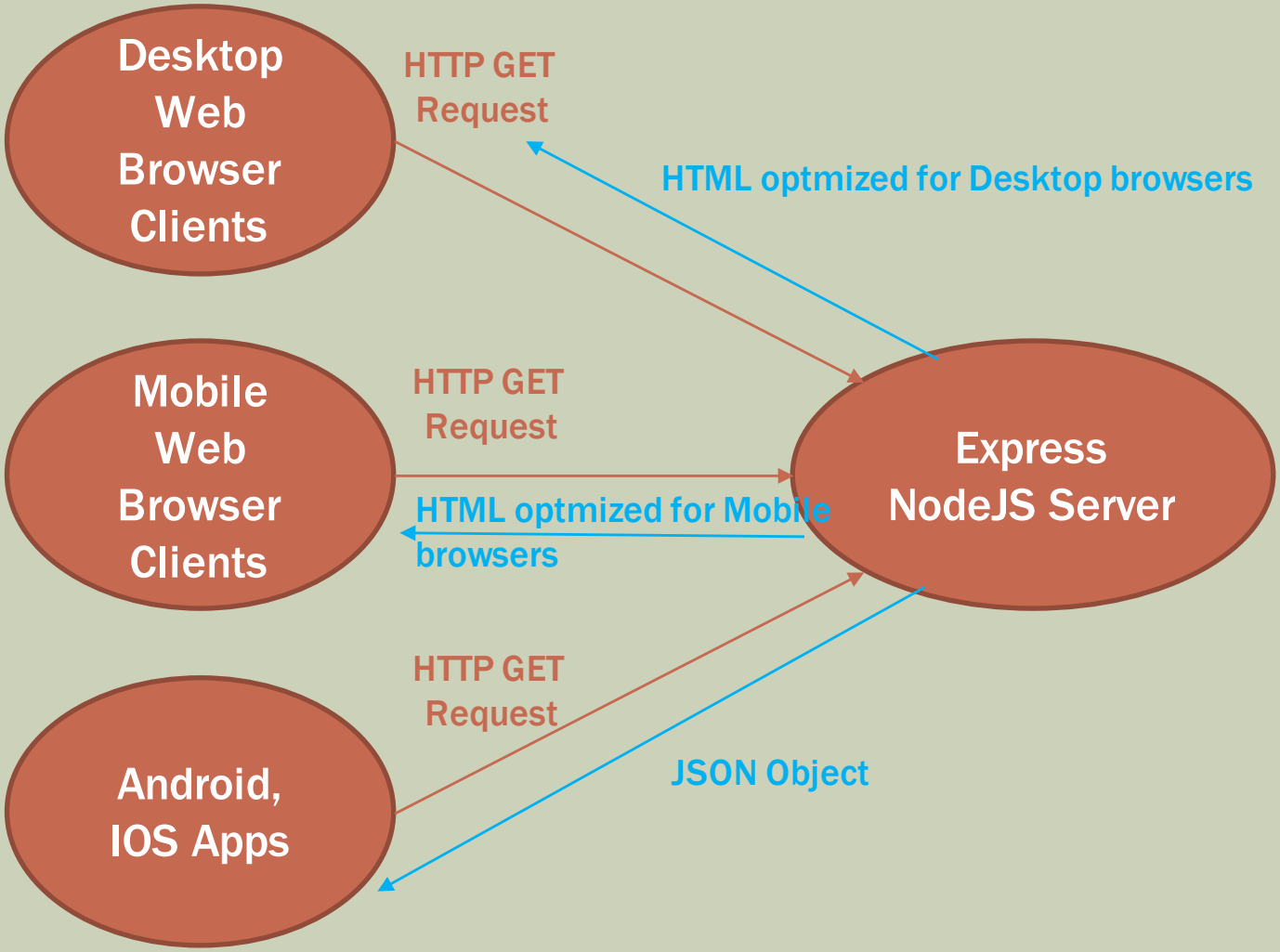
```
var express = require('express');
var app = express();

app.set('view engine', 'ejs');

app.get("/", function(req,res) {
    res.render("welcome", {greetMsg: greetingMessage()});
});

function greetingMessage() {
    var dateToday = new Date();
    var theHour = dateToday.getHours();
    if (theHour > 18) {
        gMessage = 'Good Evening';
    } else if (theHour > 12) {
        gMessage = 'Good Afternoon';
    } else {
        gMessage = 'Good Morning';
    }
    return gMessage;
}
app.listen(process.env.PORT || 8099);
```

Declare the use of EJS as 'view engine'

EJS injects value of 'greeting' into views/welcome.ejs (the view template)

Source: https://github.com/raymondwcs/ejs/blob/master/greetings/server.js

MVC EXAMPLE – MULTIPLE VIEWS



RENDERING DIFFERENT VIEWS

```
app.get("/", function(req,res) {
    console.log("User-agent : " + req.headers['user-agent']);
    if (req.headers['user-agent'].match(/Windows/) ||
        req.headers['user-agent'].match(/Macintosh/)) {
        res.render("welcome_d", {greetMsg: greetingMessage()});
    }
    else if (req.headers['user-agent'].match(/iPhone/) ||
             req.headers['user-agent'].match(/iPad/) ||
             req.headers['user-agent'].match(/Android/)) {
        res.render("welcome_m", {greetMsg: greetingMessage()});
    }
    else {
        var jobj = {gMessage: greetingMessage() };
        res.send(JSON.stringify(jobj));
    }
});
```

Check `user-agent` field in the HTTP Request Header

MVC EXAMPLE – EXPRESS CONTROLLER

```
app.get("/show", function(req,res) {
    var fields = filterResult(req.query.id);
    mongoose.connect(MONGODBURL, function(err) {
        assert.equal(err,null);
        var Kitten = mongoose.model('Kitten', kittySchema);
        Kitten.find({},fields,function(err,results) {
            assert.equal(err,null);
            db.close();
            res.render("kitties", {kitties: results});
        })
    });
});

// Controller
function filterResult(id) {
    fields = (id == "admin") ? "name age -_id" : "name -_id";
    return(fields);
}
```

/show?id=admin

Send results (an array) to views/kitties.ejs

Find critiera

Full source: https://github.com/raymondwcs/ejs/blob/master/mvcfind/server.js

MVC EXAMPLE – EJS VIEW

JS embedded in between <% %>

```
<html>
<body>
<H2>Details of all Kitties</H2>
<ol>
<% for (var i=0; i<kitties.length; i++) { %>
    <li>Name: <%= kitties[i].name %>
    <% if (kitties[i].age != null) { %>
        Age: <%= kitties[i].age %>
    <% } %>
<% } %>
</ol>
</html>
</body>
```