Routing *views and controllers dinamically* with a simple solution.
Features:
- *Dynamic routing for controllers. Controller will be found automaticaly*
- *Dynamic routing for views. Your views will be found automaticaly*
- *Infinite levels of directories for view path.*
  /app/controllerx/d1/d2/d3/view
- *Send custom parameters for controller thru url*
- Especific controller method call passing url custom params in object format

Url example:
- /app/controller1/admin/user/edit/userid=123
- /app/controller-camelcase/admin/user/edit/userid=123

This views should be:
- /app/mvc/views/controller1/admin/user/edit.html
- /app/mvc/views/controller-camelcase/admin/user/edit.html

Controllers should be:
- /app/mvc/controllers/Controller1.js
- /app/mvc/ControllerCamelcase.js

Controller Controller1 will be found automaticaly View edit.html will be found automaticaly
Controller ControllerCamelcase will be found automaticaly too, and also its name will be parsed in camelcase due to the use of "-" separator, which is an AngularJs behavior that I kept
Requirements
- *Partials must be inside /app/mvc/views folder*
- *Urls must start with app/*
- *Controller name must be right after app in url:*
  app/mvc/controllers/controllername/

So, directory organization is:
- app/
- app/mvc/
- app/mvc/controllers/
- app/mvc/views/

Obs: You dont need to work with this requirements if you're able to edit my code so feel free to change it ;D
Here is how I am doing it:
Settings:

```
var settings = {
     router: {
          default_route: '/app/dashboard/index'
     },
     path: {
```

```
        app: 'app',
        views: 'app/mvc/views/',
        controllers: 'app/mvc/controllers/',
        models: 'app/mvc/models/',
        services: 'app/mvc/services/',
        partials: 'app/partials/',
        css: 'css/',
    }
};
```

RouteProvider – This is where view is loaded dinamically

app.js

```
var app = angular.module('MyApp', ['ngRoute']);

//routing

app.config(['$routeProvider',
  function ($routeProvider) {
    $routeProvider.
     when('/'+settings.path.app+'/:name*', {
        templateUrl: function (a) {
            var name = a.name == undefined ? a :
a.name;
            var parts = a.name.split('/');
            var page = '', part;
            for (i in parts) {
                part = parts[i].split("=");
                if (part[1] != undefined)
continue;
                if (parts[i] == '') continue;
                page += parts[i] + '/';
            }
            var page = page.substr(0, page.length -
1);
            return settings.path.views + page +
'.html';
        },
        //template:' loading',
        controller: 'Router'
     }).
     otherwise({
        redirectTo: settings.router.default_route
     });
  }]);
```

Loader – Responsable for loading controller file

Loader.js

```
var Loader = {
     scripts: {},
     styles: {},
     intervals: {},
     interval_limit: 1000,
     debug: false,
     getDebugTime: function () {
          return (Loader.debugging) ? (new
Date()).getTime() : 'standard';
     },
     check: function (objectName, callback) {
          if (typeof callback != "function") return
false;
          Loader.intervals[objectName] = { interval:
null, count: 0 };
          Loader.intervals[objectName].interval =
setInterval(function () {
               if (Loader.intervals[objectName].count
== Loader.interval_limit) {

     clearInterval(Loader.intervals[objectName].inter
val);
                    return false;
               }
               if (!window[objectName]) {

     Loader.intervals[objectName].count++;
                    return false;
               }

     clearInterval(Loader.intervals[objectName].inter
val);
               callback();
               return objectName;
          }, 1);
     },
     load: {
          js: function (path) {
               path = path.replace(".js", "");
               if (Loader.scripts[path]) return;
               var script =
document.createElement("script");
               script.id = "controller-" +
```

```
path.replace(/\ /gi, ' ').replace(".js",
"").toLowerCase();
                script.src = path + ".js?version=" +
(Loader.getDebugTime());
                document.body.appendChild(script);
                Loader.scripts[path] = true;
            },
            css: function (path) {
                path = path.replace(".css", "");
                if (Loader.styles[path]) return;
                var elm =
document.createElement("link");
                elm.href = path + ".css?version=" +
(Loader.getDebugTime());
                elm.rel = "stylesheet";
                document.head.appendChild(elm);
                Loader.styles[path] = true;
            }
        }
}
```

Router – This is where controller will be loaded dinamically
Router.js

```
app.controller("Router", function ($scope, $route,
$http, $routeParams, $filter, $injector, $location) {
    var a =
window.location.hash.replace("#/app/","").split("/"),
b = a[0].split("-"), c = '', d, e = '', f, g = {}, h,
i;
    for (i in b) c += b[i].upperCaseFirst();
    d = { $scope: $scope, $http: $http,
$routeParams: $routeParams, $filter: $filter,
$location: $location };
    for (i in a) {
        if (a[i] == '') continue; h =
a[i].split("=");
        if (h[1] == undefined) { e = a[i];
continue; }
        e[h[0]] = h[1]; g[h[0]] = h[1];
    }
    d['$action'] = e;
    Loader.load.js(settings.path.controllers + c);
    Loader.check(c, function () {
        f = window[c][e];
```

```
          if (window[c].css != undefined &&
window[c].css.length > 0) {
               for (i in window[c].css)
Loader.load.css(settings.path.css +
window[c].css[i].replace(".css", "") + ".css");
          }
          $route.routes['/app/:name*'].controller =
f;
          $route.reload();
          setInterval(function () {
               $route.routes['/app/:name*'].controller
= 'Router';
          }, 200);
     });
});
```

Controller example

```
var Dashboard = {
     //this css will be loaded automatically
     css:['dashboard.min.css'],
     index: function ($scope) {
          $scope.var1 = "Testing 123123";
     },
     summary2: function ($scope) {
          $scope.var1 = "Testing 1";
     }
};
```

UpercaseFirst prototype – can use something else if wanted This is only for camelcase naming

```
String.prototype.upperCaseFirst  = function () {
    var f = this.charAt(0).toUpperCase();
    return f + this.substr(1);
}
```