Ctrl + click to follow for [Next Feature Overview]:

1. Storage Abstraction Layer Module (SALM)
   a. standalone & external to other modules
   b. every module does the same (checks if SALM object is available, if not loads it into module's scope)
   c. powered by browser's sessionStorage functionality
   d. single public method with two parameters

```
public object getModuleData(
                                        pathToConfig,
                                        moduleDOM_Object
                                    );
```

   where

   pathToConfig = '/path/to/config.txt',

   moduleDOM_Object = <what to update in module's DOM>

   e. available abstractions
      - flat file
      - database
      - service
   f. every module will contain its own module helper bound only to its parent, f.e.

            module-helper-main.js,

            module-helper-profile.js

                        .

                        .

   g. every module will have its own config file adhering to some schema (abstraction) named **config.txt**
   h. the idea: any module can be copied/pasted to new site and running with zero-configuration
   i. h() dependency : SALM script must be placed in some known place, f.e. root folder of the website, and this location has to be specified in each module's module-helper-{x}.js

config.txt definition

```
[ Flat File Abstraction ]
{
    “isFile” : true|false,
    “storageLocation” : "/path/to/upload/release_notes.txt"
}
```

```
[ Database Abstraction ]
{
    “isDatabase” : true|false,
    “serverSideScriptUrl” : "/path/to/server_scripts/release_notes_uploader.php",
    “contentType” : "text/plain | application/json | ...; charset=utf-8",
    "dataType" : "text | json | ...",
    “serverName” : "localhost",
    “portNumber” : 1234,
    “databaseName”, “some_database_name”,
    “userName” : "some_user_name",
    “userPassword” : "some_user_password",
    “queryString” : "SELECT * FROM ... JOIN ... ON ...",
    ‘databaseRequestRequiresAuth” : true|false,
    “databaseRequestAuth” : {
       “user” : "user_name",
       “password” : "user_password"
    }

}
```

```
[ Web Service Abstraction ]
{
    “isService” : true|false,
    “serviceUrl” : "localhost:8080/services/wcf/Products.svc/GetProducts",
    “contentType” : "text/plain | application/json | ...; charset=utf-8",
    "dataType" : "text | json | ...",
    “serviceMethodRequiresParams” : true|false,
    “serviceMethodParams” : {
      “param_1” : "value 1",
      “param_2” : "value 2",
                    .
      “param_n” : "value n"
    },
    “serviceMethodRequiresAuth” : true|false,
    “serviceMethodAuth” : {
      “user” : "user_name",
      “password” : "user_password"
    }
}
```

You only need to provide one abstraction for particular module, f.e.

main module may contain **Flat File Abstraction** whereas **profile** may contain **Web Service Abstraction**.

You cannot provide two abstractions for particular module, f.e. **Flat File Abstraction** and **Database Abstraction**. This restriction is put in place for optimization purposes.

Which storage abstraction is to be used is represented by the first property of each abstraction, i.e. **isFile**, **isDatabase**, **isService**.

The most complex configuration features **Web Service Abstraction**,

where you have two additional “driving properties”,

namely **serviceMethodRequiresParams** and **serviceMethodRequiresAuth**.

If any of them holds true, then subsequent properties respectively has to be provided.