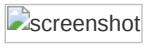
Tutorial: First Module

Edit New Page

Michael Thuy edited this page on 30 May 2015 · 6 revisions

freedom.js lets you build powerful portable web app backends without worrying about the complex distributed systems problems in a typical web framework. You can write your entire application using core client-side technologies you're already familiar with: HTML, JavaScript, and MVC frameworks. Ship the code using a static web server and we'll do the rest!

We'll start with making a simple Hello World app: Counter. The source code for this tutorial can be found in this repository.



Declare your Backend Module

The very first thing we'll need to write is a *manifest file*, which we will name `manifest.json`. The manifest file is just a JSON-formatted table that describes the backend module's name, description, version number, etc. At a high-level, it tells *freedom.js* where the code for the module is and what permissions it requires.

```
{
  "name": "Counter",
  "description": "Counter Sample FreeDOM application",
  "app": {
    "script": "counter.js"
  }
}
```

Most of the file is self-explanatory. The "app" attribute points to the script that contains the code for the backend module. Note that this path is relative to the manifest file.

Start writing your backend module

Let's start with adding a log statement to `counter.js`. *freedom.js modules* have access to console.log() and console.error().

```
console.log('Hello World!');
```

Write your page

Now we have to write our main page, `index.html`. This outer page instantiates *freedom.js* and all of the backend modules in the application (currently there is only one). We can instantiate *freedom.js* by including the script and pointing the *data-manifest* attribute to our *manifest file*.

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<script type='text/javascript'
    src='/freedom.js'
    data-manifest='manifest.json'>
  {
    "strongIsolation": true,
    "debug": true,
    "stayLocal": true
  }
</script>
...
```

Pages 36

Find a Page…

Home
Applications
Common Issues
Compiling freedom.js
Creating a Social Provider
Debugging
Debugging Script Parse Errors
Existing API Providers
FAQ
freedom.js OAuth
freedom.js structure
freedom.js structure: Consumer Interface
freedom.js structure: module cleanup
freedom.js structure: Module Environment
freedom.js structure: Provider Interface

Show 21 more pages…

Clone this wiki locally

https://github.com/freedc

```
</head>
<body>
...
```

In the text content of the script tag, we can specify a JSON object of options. Currently there are 3: **strongIsolation** should be true for all production-ready code. During development, it can help to occasionally turn this off for better error messages. **debug** prints internal *freedom.js* log messages to console. **stayLocal** is a highly experimental feature, which we currently recommend to stay true.

Remember to point the script tag to the proper paths! The paths above reflect the configuration when a web server is serving from the base directory of the `freedom.js` repository.

You can also find a compiled version of the library here or compile it yourself from this repository.

Run it!

The easiest way to run the demo is by cloning the `freedom.js` repository. From the base directory, run:

```
grunt demo
```

Then navigate to **http://localhost:8000/demo/counter** on your browser. This will compile *freedom.js* and start a local web server (note that freedom.js apps cannot be loaded from file:// due to various security policies). If you want to do this yourself, remember to point your `index.html` script tags to the right place.

Next Page

1. Message Passing - Adding message passing interfaces