Jellyfish Theme API

Hey! If you've wanted to make a custom UI but know HTML or don't want to go through the hassle of manually supporting exploits? Well, with the Jellyfish theme system, you can create a custom UI in HTML, and instantly push to Windows & Mac users of many different exploits!

Getting Started

Examples

If you'd rather just download an example theme and modify it to your liking, you can use jellyfish-ui, the default theme or [erminal] (https://github.com/jellyfish-lsef/erminal)

Folder Structure

You'll need to create a folder (and in the future, a GitHub repository) to store your theme. Every theme needs a `package.json` file, which includes the metadata on what your theme is. Copy and paste the template below into your package.json in the theme's folder.

```
{
  "name": "My Great Theme",
  "description": "A amazing theme for Jellyfish",
  "author": "Me",
  "version": "1.0.0",
  "main": "index.html",
  "keywords": [
    "jellyfish-ui"
  ]
}
```

This is the base template that includes everything a Jellyfish theme requires. You can add some optional keys to specify even more about your theme.

```
{
  "name": "My Great Theme",
  "description": "A amazing theme for Jellyfish",
  "author": "Me",
  "version": "1.0.0",
  "main": "index.html",
  "keywords": [
    "jellyfish-ui"
  ],

  width: 1000,
```

```
  height: 500,
  fixedSize: true,
  borederless: true
}
```

These keys define even more about your theme. `width` and `height` define the default size of the window, `fixedSize` defines whether or not the window can be resized, and `borderless` defines if the window has a window frame.

The actual page

To have Jellyfish load a theme, there has to be, well, a theme for it to load. You can name the HTML file whatever you want, as long as it corresponds to the `main` key that we specified earlier in the package.json, by default, this is `index.html`, so create that file and copy and paste the below in.

```
<html>
  <head>
    <title>My Amazing Theme</title>
    <meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=no" />
    <!-- without specifying charset monaco goes Â-->
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <style>
      ::-webkit-scrollbar {
        display: none;
      }
    </style>
  </head>
  <body style="margin: 0;">
    <script> window.addEventListener("load",jellyfish.init) </script>
  </body>
</html>
```

This is the basics of what we need to get a Jellyfish theme up and running. You may remove the script tag at the bottom and call `jellyfish.init` when your code has finished loading.

Loading our theme in Jellyfish

You might actually want to use and test your theme, rather than just coding it. That's great. To load your theme in Jellyfish, simply go to the Tools tab (on the default theme) and change `jellyfish-lsef/jellyfish-ui` to your theme ID and click `Change theme`. Restart Jellyfish and your theme should load.

Hold on, what's a theme ID?

A theme ID is a identifier that tells Jellyfish where your theme is. In production, you'll use your GitHub repo's identifier, for example `thelmgn/my-amazing-theme`, however when you're developing your theme, you might not want to have to push your changes every time. To do that, simply put the path to your theme prefixed by `local/`, i.e. `local//Users/thelmgn/Documents/my-amazing-theme`

Debugging my theme

To debug your theme, press `Control (or Option) + Shift + I` at the same time to show the Chrome devtools. On Windows, the shortcut is `F12`. You can use `Ctrl+R` to reload your theme without restarting Jellyfish.

Have fun!

Jellyfish API

You might be thinking, that's great, but how do i interface with Jellyfish? Glad you asked, when you need to interface with Jellyfish, you can use the `jellyfish` object, and when Jellyfish needs to interface with you, it'll call a function you define on the window, for example

```
window.gotExploit = function() { document.title = "Jellyfish for " +
jellyfish.exploitName }
```

A list of all the properties, functions and events you can use is listed below.

Please note, that if Jellyfish updates, this document may be out of date. Please see preload.js for the latest docs.

Properties

```
string jellyfish.version = "2.0.0"
```

This defines the Jellyfish version, in the semver format.

```
string jellyfish.platform = ("win32","darwin")
```

This specifies what OS Jellyfish is running on.

```
string jellyfish.exploit =
("loading","null","calamari","sirhurt","synx","fluxus","wrd","easyexploits")
```

This defines the exploit ID. Note, that the listed above exploits are not all the exploits Jellyfish will support in the future, use `jellyfish.supportedExploits` to get the exploits available on this system, and the `jellyfish.exploits` dictionary to get all exploits in the current version of Jellyfish.

```
string jellyfish.exploitName = ("Loading","nullExploit","Calamari-
M","SirHurt","Synapse X","Fluxus","WRD","EasyExploits")
```

This defines the exploit's friendly name. Note, that the listed above exploits are not all the exploits Jellyfish will support in the future, use `jellyfish.supportedExploits` to get the exploits available on this system, and the `jellyfish.exploits` dictionary to get all exploits in the current version of Jellyfish.

```
string[] jellyfish.supportedExploits = (
  ["calamari","fluxus"],
  ["sirhurt","synx","fluxus","wrd","easyexploits"]
)
```

This defines the exploits supported on the current system. Note, that the listed above exploits are not all the exploits Jellyfish will support in the future.

```
dict[string: string] jellyfish.exploits = {
    null: "nullExploit",
    calamari: "Calamari-M",
    sirhurt: "SirHurt",
    synx: "Synapse X",
    fluxus: "Fluxus",
    wrd: "WRD",
    easyexploits: "EasyExploits"
}
```

This defines the exploits supported on the current Jellyfish version. Note, that the listed above exploits are not all the exploits Jellyfish will support in the future.

```
string jellyfish.datadir = "/Users/thelmgn/Documents/Jellyfish"
```

This defines the Jellyfish data dir.

Functions

```
void jellyfish.init()
```

This will initialize some functions and fire some events.

```
void jellyfish.inject(bool argument)
```

This will tell the current exploit to inject. It is expected that no other inject calls can be made until `enableInjectBtn` is called.

```
void jellyfish.runScript(string script)
```

Runs a script using the selected exploit.

```
void jellyfish.startCrawl()
```

This will start a crawl for every script in the scripts folder on disk. If this is called more than once, the previous operation is cancelled.

```
void jellyfish.startRemoteCrawl()
```

This is the same as `jellyfish.startCrawl` but for the Jellyfish `More Scripts` remote script hub.

```
void jellyfish.getScript(string filename, function(error,script) callback)
```

This will read a file from disk, or if the filename starts with `jellyapi:` download a string from the internet, calling the callback with two arguments, `error` and `script`. If the file is on disk, this will only succeed if the file path is a child of `jellyfish.datadir`

```
void jellyfish.saveScript(string script)
```

Opens a save dialog to save a file ending with `.lua` or `.txt`.

```
string jellyfish.joinPath(string[] pieces)
```

This calls `path.join`, and joins all the pieces into one path, corresponding to the OS' method of path delimitation.

```
void jellyfish.setTopmost(bool topMost)
```

This will set the current window to show above all other windows or not.

```
void jellyfish.setExploit(string exploit)
```

Select the exploit by it's ID. Jellyfish will restart after this is called.

```
void jellyfish.setTheme(string themeId)
```

Change the current theme. If the themeId is anything except `jellyfish-lsef/jellyfish-ui`, Jellyfish will validate that this is a valid theme, then ask the user if they want to change theme. If the user accepts, Jellyfish will restart.

```
void jellyfish.attemptLogin(string username, string password)
```

This will attempt a login when the user enters credentials. Do **not** hide a login page as the result of this finishing, wait for `loginSuccess` to be called.

Events

Events are functions that get called by Jellyfish to communicate with your theme. Events can be defined as simply

```
function eventName(arguments) {
 // Code here
}
```

```
injectionStatusChange(string text)
```

This is called when the current status of the injection changes, show this to the user **and disable any injection buttons**.

```
enableInjectBtn()
```

This is called when the injection is complete and it is safe to clear the status and reenable the injection buttons.

```
onScriptRun()
```

This is called when Jellyfish detects a script has been ran. There are no guarantee that this will be called or that every call of this is user initiated. Use only for asthetics.

```
gotExploit()
```

This is called when `jellyfish.exploit` and `jellyfish.exploitName` have been changed.

```
gotSupportedExploits(string[] exploits)
```

This is called when `jellyfish.supportedExploits` changes.

```
createScript(Object path, string fullPath)
```

This is called when a script is found either on the script hub, or on the local disk. **When this is called by the remote script hub, only base and dir are defined on the parsed path**, and the fullPath will always start with `jellyapi:`. For more information on the parsed path, see https://nodejs.org/api/path.html#path_path_parse_path

```
crawlFinished()
```

This is called when the script hub crawl operation completes. This isn't sent when using the remote script hub [bug].

```
apiConnected()
```

This is called when a connection to the remote script hub is made, and it is safe to show the script hub button and call `startRemoteCrawl`

```
apiDisconnect()
```

This is called when we lose connection to the remote script hub and it is no longer safe to call `startRemoteCrawl`

```
showLogin(string username, string password)
```

This is called when authentication fails. The current username and password is provided to prefill input boxes.

```
loginSuccess()
```

This is called when the login box can be hidden.