Neriven API – How to build an app

Inside app.zip

An app is just a regular zip file. It usually consists of 3 or more files.

Send us your file to app@neriven.com

manifest.json (required)

Here's an example:

```
{
  "id": "bc594b1b-310e-48c9-98fe-aa5b32298b69",
  "name": "Gmail",
  "description": "Gmail is a free, advertising-supported email service developed by Google. Users can access Gmail on the web and through the mobile apps for Android and iOS, as well as through third-party programs that synchronize email content through POP or IMAP protocols.",
  "category": "Email",
  "small_icon": "Gmail.svg",
  "large_icon": "Gmail.svg"
}
```

- id (required) should be a unique, valid UUID.
- name (required)
- description (required)
- category (required)
- small_icon (required) — relative path to an icon which will be shown in the toolbar.
- large_icon (required) — relative path to an icon which will be shown on the marketplace.
- is_marketplace (private) — this enables an app to access a list of installed applications. This is required by Marketplace app.
- is_plus (private) — adds a save button which trigger a private __save event.
- autorun (private) — by default all applications are lazy-loaded, meaning that they will load after a user implicitly activates it. This allows an app to be loaded in the background, right after Tilez start.

main.js (required)

main.js executes when an app starts.

Below is an example of a very basic app. It creates a tile, opens Google in it and binds a callback for dom_ready event.

```
var mainTile = app.addTile();
mainTile.setURL('https://google.com/', () => {
   alert('Page is loaded');
});
```

Icon(s) (required)

SVG is preferred, PNG is acceptable.

Static files

Other files are accessible via HTTP. For example you can package an HTML file inside zip and open it in a new tile like this:

```
app.addTile().setURL(app.contents_prefix + 'file.html')
```

API

App context and tile context

Functions and event listeners could be executed either in the app context or the tile context. There are a few things to keep in mind:

1. Contexts are isolated from each other. You can not access a variable declared in one context inside the other.
2. Both context have predefined app variable which is an instance of the App class.
3. The tile context has predefined tile variable which is an instance of the Tile class.
4. Event listeners that are bound to a tile run in tile context.
5. Default context in main.js is the app context.
6. You can access a webpage contents inside tile context. This is injected directly to a web page meaning that you have full access to window, document and other global objects.

Here's an example:

```
// (5) main.js runs within app context
var greeting = 'Hello!';
alert(greeting);  // (1) This works
var mainTile = app.addTile()  // (2) app is available everywhere

mainTile.on('dom_ready', () => {
   // (4) Event listener for tile's event listener runs in the tile context
```

```
    alert(greeting);  // (1) undefined — greeting is not available here.
    alert(mainTile);  // (1) also undefined
    alert(tile); // (3) this works
    alert(window.location.href);  // (6) we can access window
    alert(document.getElementsByTagName('title')[0].innerText);  // (6) DOM is also available here
    app.trigger('greet');  // (2) app object is available in the tile context...
});

// ...as well as in the app context
app.on('greet', () => {
    alert(greeting);  // (1) this works
});

mainTile.setURL('http://initech.co.il/');
```

App class

Properties

id

An UUID of the app. Example: e870974d-18f4-466a-8dd5-7513a1d45a0d

marketplace_url

Contains an URL of the Marketplace web interface. Example: http://139.59.144.78

contents_prefix

This is a path under which static files is located. Example: http://139.59.144.78/apps/e870974d-18f4-466a-8dd5-7513a1d45a0d/code/ See also: Static files.

Methods

on(event, listener)

Binds an event listener. See the list of the available events below, and you can also use your own custom events. Example:

```
app.on('restore', () => {
    alert('Tilez main window was restored');
});
```

Although you can bind app events within both app and tile contexts, it is recommended to use app context for binding events.

trigger(event, params)

Triggers an arbitrary app event, causing all listeners bound to this event to fire. The second param is passed to the event listener. Example:

```
app.on('custom_event', (action) => {
   alert('Custom event happened! And it ' + action);
});
app.trigger('custom_event', 'rocks');
```

activate()

Causes an app to activate itself.

addTile(size=1)

Adds a new tile, returns an instance of Tile class (see below). Example:

```
var mainTile = app.addTile(2);
mainTile.setURL('http://initech.co.il/');
```

setUserAgent(userAgent, width)

Changes default user agent and a viewport width. Example:

```
app.setUserAgent('Tiles/1.0', 800);
```

hide() and show()

Causes an app to disappear and appear again in the app bar.

Events

open

When a user switches to an app using the app bar.

window_minimize and window_restore

Tilez main window is minimized or restored.

Tile class

Methods

on(event, listener)

Binds an event listener. See the list of the available events below. You cannot use your own custom events. Example:

```
app.on('back', () => {
   alert('It works');
});
```

All listeners are executed within tile context.

setURL(url, onDomReady = null)

Loads a webpage with the specified url into tile. The optional second argument is an event listener that automatically binds to dom_ready.

getURL()

Returns a URL of a currenly loaded web page.

remove()

Removes a tile.

maximize()

Sends a tile to main view.

minimize()

Returns a tile back to tiles band.

setLocked(value)

Makes a tile uncloseable and unminimizeable.

setPrimary(value)

Makes a tile uncloseable.

activate()

If tile is maximized, then it emulates a click on tab.

Events

dom_ready

A tile has loaded a page.

back

A user have clicked "Back" button.

maximize

Tile was maximized.

minimize

Tile was minimized.