Setup / Getting started

- Explanation of what Node Webkit is
- Advantage
- Limitations

Download NW.js developer version and unzip it.

Hello World

Create a file named package.json and copy the following code into it. Save it in the Node Webkit folder.

```
{
    "main": "app/index.html",
    "name": "app",
     "version": "1.0"
}
```

Create a folder named “app” in the Node Webkit folder and save a file named index.html in it with the following code.

```
<!DOCTYPE html>
<html>
      <head>
            <meta charset="UTF-8">
            <title></title>
```

```
            <script src="js/app.js"></script>
            <link rel="stylesheet" href="css/app.css">
      </head>
      <!--_____________________________________________________BEGIN
APP-->
      <body>
<h1>Hello World. This is an empty app </h1>


      </body>
      <!--_____________________________________________________END
APP-->
</html>
```

On windows you can click the file nw.exe to see the empty app

On Mac you can download a module here: `sudo npm install -g nodewebkit`

This will allow you to launch the app by going to the SDK directory from the command line and typing: `nodemon .`

Live Reload

```
var path = './app';
var fs = require('fs');

fs.watch(path, function() {
    if (location){
        location.reload();
```

```
    }
});
```

Developer tools

To see Chrome Developer tools you can right click and choose it in the menu or F12

Setting default window size

```
{
    "main": "app/index.html",
    "name": "app",
    “version”:”1.0”,
    "window": {
        "title": "app",
        "width": 1200,
        "height": 200,
        "min_width": 400,
        "min_height": 200,
        "max_width": 800,
        "max_height": 600,
        "resizable": false
    }

}
```

A working app example

Place the todos folder in the node app and modify the package.json file “main” property to reference the index file inside the todos folder.

```
{
    "main": "todos/index.html",
    "name": "app",

   //. . .rest of code
}
```

Changing the top left corner and menu icon

In the Node Webkit folder place the icon you want in PNG format and reference it in the package.json file.

```
{
    "main": "app/index.html",
    "name": "app",
    "version": "1",
    "window": {
        "title": "app",
        "icon": "link.png",
        "width": 1200,
        "height": 200,
        "min_width": 400,
        "min_height": 800,
        "max_width": 800,
        "max_height": 600
```

```
    }

}
```

Packaging the app

To package the app you can use Node Webkit Builder. Install it with this command:

```
npm install nw-builder -g
```

In your terminal CD into the node webkit directory and run the following command:

```
nwbuild -v (version of node webkit) -p win64,osx64,linux64 .
```

Example:

```
nwbuild -v 0.16.1 -p win64,osx64,linux64 .
```

The platforms can be `'win32', 'win64', 'osx32', 'osx64', 'linux32', 'linux64'`

Note:

When nw-builder is run with the version 0.15.1 of node webkit it throws an error. The earliest it seems to work is :

```
nwbuild -v 0.12.3 -p win64,osx64,linux64 .
```

You will now have a folder named “build” in the node webkit folder that contains the packaged app(s).

Removing the toolbar

When you package your app you might see the Chrome toolbar after running the packaged executable. If so you can remove it by first typing the following into your package.json file in the “windows” object prior to running nw-builder.

```
"toolbar": false,
```

Changing the executable icon (Windows OS)

First, convert an image to an ico file:

http://icoconvert.com/

Download and install Resource Hacker:

http://www.angusj.com/resourcehacker/

Open resource hacker and file/open the .exe file in your build folder

Under the icon menu select the start and choose “open file with new icon”.

In the window that appears select the new icon and then click the “replace” button.

Go to “file” and “save”.

Update the cache (Windows)

C:\Users**USERNAME**\AppData\Local\Microsoft\Windows\Explorer

Other operating systems: https://github.com/nwjs/nw.js/wiki/Icons

Creating Menus

To create menus you must use the nw.gui module and first create the “menubar”

This will be empty by default.

```
var gui = require('nw.gui');
var win = gui.Window.get();

var menubar = new gui.Menu({
    type: 'menubar'
});

/* for mac
      menuBar.createMacBuiltin(“app“);
*/

win.menu = menubar;
```

Adding Menus

```
var gui = require('nw.gui');

var win = gui.Window.get();

var menubar = new gui.Menu({
    type: 'menubar'
});

menubar.append(new gui.MenuItem({
```

```
    label: 'First Menu'
}));

win.menu = menubar;
```

Adding sub-menus

```
var gui = require('nw.gui');

        var win = gui.Window.get();

        var menubar = new gui.Menu({
            type: 'menubar'
        });

        var subMenuSpace = new gui.Menu();

        subMenuSpace .append(new gui.MenuItem({
            label: 'Item A'
        }));

        subMenuSpace .append(new gui.MenuItem({
            label: 'Item B'
        }));

        menubar.append(new gui.MenuItem({
            label: 'First Menu',
            submenu: subMenuSpace
        }));
```

```
win.menu = menubar;
```

Add event listeners to menu items

```
subMenuSpace .append(new gui.MenuItem({
    label: 'Item A',
    click:function(){
    alert(" you clicked me")
    }
}));
```

Modifier keys

```
subMenuSpace.append(new gui.MenuItem({
      label: "Item A",
      type: "checkbox",
      icon:"thing.png",
      key:"M",
      modifiers:"ctrl-shift",
          click:function(){
            alert("you clicked me")
          }
}));
```

Open a new window

```
subMenuSpace.append(new gui.MenuItem({
    label: "Item A",
    type: "checkbox",
    icon: "thing.png",
    tooltip: "Hello World",
    key: "M",
    modifiers: "ctrl-shift",
    click: function() {
        window.open('empty.html', {
            "position": "center",
```

```
            "focus": true,
            "toolbar": false
        });
    }
}));
```

Toggling windows using JQuery load() function

If you want to bypass using the Node Webkit menus and simply create your own you can use JQuery.load() (see example in code assets).

Converting a Node.js/Express web app to desktop app

You can convert a simple Node/Express app (that does NOT include a database) to a Node Webkit app that includes a view engine.

The following steps show how this is possible.

Install Node.js

Install express (npm install express) and a view engine (npm install handlebars-express)

Create a folder named app and a folder named views

Create your package.json file:

```
{
      "name": "app",
      "main": "app/index.html",
      "version":"1.0"
}
```

Go to the app folder and create a file named **index.html**. Inside of this copy the following code:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<!--_____________________________________________________BEGIN APP-->
<body>
    <script>
    </script>
    <script>
    var express = require('express');
    var app = express();
    var expressHbs = require('express-handlebars');
    app.engine('hbs', expressHbs({
        extname: 'hbs'
    }));
    app.set('view engine', 'hbs');
    app.get("/", function(req, res) {
        res.render("index", {
            item: "weeeeeeeee"
        })
    });
    app.listen("3000", function(err) {
        if (err) {
```

```
            console.log("server is not working");
        } else {
            console.log("Server is working on 3000");
        }
    });

    window.location.href = 'http://localhost:3000';
    </script>
</body>
<!--_____________________________________________________END APP-->
</html>
```

Go to the views folder and create a new file called index.hbs. Inside this file copy the following code:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>

    </head>
    <!--_____________________________________________________BEGIN APP-->
    <body>
        <p>Oink</p>
        {{item}}
    </body>
    <!--_____________________________________________________END APP-->
```

```
</html>
```