Electron

Desktop-Applications with JavaScript



Thanks to Olaf and Peter to host this meetup!



- OpenSource
- WebGIS



- Products
- Consulting
- Development
- Support



Customers

- Deutsche Bahn
- Vattenfall
- Administration



Products

- Mapbender
- Metador
- Mops



Involvement

- FOSSGIS
- FOSS4G
- OSGeo
- OGC



Locations

- Bonn
- Berlin
- Freiburg



By the way

We are hiring

www.wheregroup.com/de/jobs_karriere



Speaker

Arne Schubert

- Head of development WhereGroup
- Local-Team FOSSGIS Conference 2018 Bonn
- OSGeo Charter Member
- Maintainer of the Node.JS showcase application YAGA



Topic

Electron

Build Desktop Applications with Node.JS



Pre-Requirements

• Node.js



Pre-Requirements Ubuntu and Debian

curl -sL https://deb.nodesource.com/setup_9.x | sudo -E bash sudo apt-get install -y nodejs



Application development with JavaScript

An Overview



Classic

Browser and server

- Browser runs JavaScript
- Server just have to communicate over HTTP protocol
- Communication over HTTP requests

It is "Just a browser". Share a window with other sites.
Window is not fully configurable



Mobile devices

Cordova

- WebView runs JavaScript
- Excerpts Plugins!
- Plugins are platform specific
- Every platform needs a different language
- Communication over an IPC or HTTP requests to external servers



Desktop

Electron

- Main-Thread is running on Node.JS
- Renderer-Thread in a Chromium-like WebView
- Both run with JavaScript
- Communication over an IPC or HTTP requests to servers



Desktop Electron

- GitHub
- Documentation



First steps - a simple Web-App Initialize Project

```
mkdir mapbender-desktop
cd mapbender-desktop
npm init # Answer the CLI. Use "electron/index.js" as entry point...
npm install --save-dev electron # add electron also as peer dependence
```



Edit package.json

Just the important lines

package.json

```
{
    "main": "electron/index.js",
    "scripts": {
        "start": "electron ./"
},
    "devDependencies": {
        "electron": "^1.7.11"
},
    "peerDependencies": {
        "electron": "^1.7.11"
}
}
```



Import dependencies

```
const app = require("electron").app;
const BrowserWindow = require("electron").BrowserWindow;
```



Create a reference for your window

electron/index.js

// Keep a global reference of the window object, if you don't, the wi
// be closed automatically when the JavaScript object is garbage coll
let win;



Write createWindow function

```
function createWindow () {
    // Create the browser window.
   win = new BrowserWindow({width: 1000, height: 800});
    // and load the index.html of the app.
   win.loadURL("https://demo.mapbender3.org/");
    // Open the DevTools.
    // win.webContents.openDevTools();
    // Emitted when the window is closed.
   win.on('closed', () => {
        // Dereference the window object, usually you would store win
        // in an array if your app supports multi windows, this is the
        // when you should delete the corresponding element.
        win = null;
```

Register default EventListeners

```
// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', createWindow);

// Quit when all windows are closed.
app.on('window-all-closed', () => {
    app.quit();
});
```

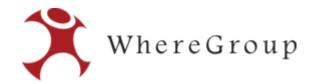


Enhance EventListeners for usage on macs

```
// Quit when all windows are closed.
app.on('window-all-closed', () => { // Notice: Rewrite!!!
    // On macOS it is common for applications and their menu bar
    // to stay active until the user quits explicitly with Cmd + Q
    if (process.platform !== 'darwin') {
        app.quit();
});
app.on('activate', () => {
    // On macOS it's common to re-create a window in the app when the
    // dock icon is clicked and there are no other windows open.
    if (win === null) {
        createWindow();
```

Start the app

npm start



A local application Initialize Project

Initialize the project according the previous web-app



Base directory for the renderer

mkdir www



Write HTML index

www/index.html

```
<html>
<head><title>Local App with Electron</title></head>
<body>
   This app is created with
        <a href="https://github.com/electron/electron/">Electron</a>.
</body>
</html>
```



Enhance dependencies

```
const path = require("path");
const url = require("url");
```



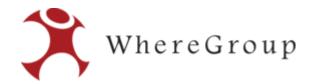
Load created HTML file in WebView

```
function createWindow () {
    // ...
    win.loadURL(url.format({
        pathname: path.join(__dirname, '../www/index.html'),
        protocol: 'file:',
        slashes: true
    }));
    // ...
}
```



Start the app

npm start



Let's style the local application Initialize Project

Initialize the project according the previous local-app



Install your Web-UI of choice

Example for bootstrap

npm install --save bootstrap popper.js jquery-slim



Enhance HTML index

Load additional dependencies

www/index.html



Enhance HTML index

Rewrite the body

www/index.html



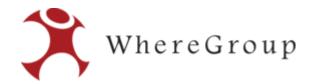
Start the app

npm start



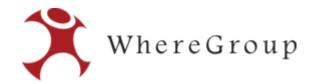
Use IPC Channels

Idea: A file browser for images



Initialize Project

Initialize the project according the previous styled-app



Directory information service

Promised based module to get a summary for a directory

TL;DR

electron/directory-info.js



Directory information service

Interface of electron/directory-info.js response

```
interface IDirectoryInfo {
    root: string;
    dir: string;
    base: string;
    ext: string;
    name: string;
    type: "file" | "image" | "directory";
}
```



Directory information service

Example respone of electron/directory-info.js

```
[{
    "root": "/",
    "dir": "/path/to/dir",
    "base": "package.json",
    "ext": ".json",
    "name": "package",
    "type": "file"
}]
```



Implement IPC on main thread

Add ipc-listner

electron/index.js

```
const ipcMain = require("electron").ipcMain;
const directoryInfo = require("./directory-info").directoryInfo;

ipcMain.on("list-directory", (context, { path }) => {
    path = path || process.cwd();
    directoryInfo(path).then((entries) => {
        context.sender.send("list-directory-response", { path, entrie }).catch((err) => {
        context.sender.send("list-directory-error", err);
    });
});
```



Enhance HTML

Restructure body of HTML

www/index.html



Create a simple templating engine for the renderer

Breadcrumb menu

www/templates.js

```
function breadCrumb(path) {
    return path.split("/").map((dirname, index, arr) => {
        if (index === 0) {
            dirname = "/";
        }
        return `<a href="#" data-path="${
        });
}</pre>
```



Create a simple templating engine for the renderer

Generic list

www/templates.js

```
function list(entries, type) {
    if (type) {
        entires = entries.filter((entry) => { return type === entry.t
    }
    return entires.map((fileEntry) => {
        return `<a href="#" data-path="${
      });
}</pre>
```



Write IPC listeners on renderer

Load dependencies

www/ipc-listner.js

```
const ipcRenderer = require("electron").ipcRenderer;
const breadCrumbTemplate = require("./templates").breadCrumb;
const listTemplate = require("./templates").list;
```



Write IPC listeners on renderer

Register listeners and send initial request

www/ipc-listner.js

```
ipcRenderer.on("list-directory-response", (context, response) => {
    document.getElementById("breadcrumb")
        .innerHTML = breadCrumbTemplate(response.path).join("");
    document.getElementById("subfolders")
        .innerHTML = listTemplate(response.entries, "directory").join
    document.getElementById("images")
        .innerHTML = listTemplate(response.entries, "image").join("")
});
ipcRenderer.send("list-directory", {});
```



Write DOM listeners on renderer

Register listeners and send initial request

www/event-listner.js

```
const $ = require("jquery-slim");
const ipcRenderer = require("electron").ipcRenderer;

$(document).on("click", "#breadcrumb a, #subfolders a", (event) => {
    ipcRenderer.send("list-directory", { path: event.target.getAttrik});

$(document).on("click", "#images a", (event) => {
    ipcRenderer.send("show-file", { path: event.target.getAttribute('}));
```



Create JavaScript index for renderer

Load additional dependencies

www/index.js

```
require("./ipc-listener");
require("./event-listener");
```



Enhance HTML index

Load renderer application

www/index.html

```
<head>
     <!-- ... -->
     <script src="index.js"></script>
</head>
```



Add an image preview

Open another window from main thread

electron/index.js

```
function createFilePreviewWindow (srcPath) {
    win = new BrowserWindow({width: 400, height: 300});
    win.loadURL(url.format({
        pathname: srcPath,
        protocol: 'file:',
        slashes: true
    }));
    win.setAlwaysOnTop(true, "modal-panel");
    win.on('closed', () => {
        win = null;
    });
}
```



Add an image preview

Register on IPC channel

electron/index.js

```
ipcMain.on("show-file", (context, { path }) => {
    createFilePreviewWindow(path);
});
```



Start the app

npm start



Build your application Install packager

npm install --save-dev electron-packager



Synopsis of the packager

```
npx electron-packager
     <location of project>
     <name of project>
      <platform>
          <architecture>
            <electron version>
            <optional options>
```



Enhance script tasks in package.json

just the important parts for this step

```
{
   "scripts": {
     "build": "rm -Rf dist && mkdir -p dist && cd dist && electron-pace
   }
}
```



Build the app

npm run build



Summary Pros

- Build for every desktop OS
- ... only with Node.js
- Common-JS ready in both threads (require(...);)



Summary

Study the architecture of IPC pattern!



Last but not least

Checkout our Twitter - Accounts

@wheregroup

@yagajs



Last but not least

We hope you enjoy our meetup!

meetup@wheregroup.com



Let's discuss now

... while drinking some beer or limonade 69!

meetup@wheregroup.com

