eBook Gratuit

APPRENEZ electron

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#electron

Table des matières

À propos 1

Chapitre 1: Commencer avec l'électron 2

Remarques 2

Qu'est ce que l'électron? 2

Applications construites sur Electron 2

Versions 2

Examples 3

Installation d'Electron 3

Les dépendances 3

Comment l'installer? 3

Bonjour le monde! 3

Installer 3

Le processus principal 4

Modèle HTML et processus de rendu 4

Lancer l'application 5

Avec electron-prebuilt installé globalement 5

Méthode 2 - Sans electron-prebuilt installé globalement 5

Chapitre 2: App plateau électronique 7

Examples 7

App plateau électronique 7

Chapitre 3: électron-winstaller 8

Introduction 8

Syntaxe 8

Paramètres 8

Examples 9

Construire JS 9

Chapitre 4: Emballage d'une application électronique 10

Introduction 10

Syntaxe 10

Paramètres 10
Examples 10
Installation du conditionneur d'électrons 10
Emballage de CLI 11
Emballage de script 11
Création de scripts npm pour automatiser le conditionnement Electron 11
Chapitre 5: Fonction à distance - utilisez les fonctions électroniques en JavaScript 13
Introduction 13
Syntaxe 13
Examples 13
Utilisation de la télécommande en définissant la barre de progression 13
Utilisation de la télécommande en définissant la fenêtre en plein écran 13
Chapitre 6: Processus principal et rendu. 14
Remarques 14
Examples 14
Communication IPC asynchrone 14
Module RMI à distance 15
Communication IPC synchrone 15
Chapitre 7: Utiliser le bootstrap en électron 17
Introduction 17
Examples 17
Relier l'électron au bootstrap 17
Chapitre 8: Utiliser nedb en électron 18
Examples 18
Installation de nedb 18
Application électronique de connexion avec Nedb 18
Insérer des données dans nedb 18
Rechercher dans nedb 18
Supprimer dans nedb 19
Crédits 20

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: electron

It is an unofficial and free electron ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official electron.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec l'électron

Remarques

Qu'est ce que l'électron?

Electron est un **framework open-source** , utilisé pour créer des applications bureautiques utilisant **HTML** , **CSS** et **JavaScript** . À l'intérieur, cela fonctionne grâce à **Chromium** et **Node.js.**

Son créateur original, GitHub , travaille avec une large communauté de développeurs pour assurer la maintenance du projet, qui peut être trouvé ici .

L'un des principaux avantages de l'utilisation d'Electron est que, étant basé sur les technologies Web, il est **multi-plateforme** , ce qui permet de déployer des applications pour Linux, MacOS et Windows, avec le même code.

Il comporte également des éléments natifs tels que des menus et des notifications, ainsi que des outils de développement utiles pour le débogage et les rapports de panne.

Applications construites sur Electron

Voici quelques exemples d'applications utilisant ce framework:

- Atome
- Slack pour le bureau
- Code Visual Studio
- GitBook
- Malédiction
- Wordpress pour Desktop

... et beaucoup d'autres .

Versions

Version	Remarques	Date de sortie
1.0.0		2016-05-09
1.0.1		2016-05-11
1.0.2		2016-05-13
1.1.0		2016-05-13

Version	Remarques	Date de sortie
1.1.1		2016-05-20
1.1.2		2016-05-24
1.1.3		2016-05-25
1.2.0		2016-05-26
1.2.1		2016-06-01
1.2.2		2016-06-08
1.2.3	Il y a plus entre ceci et 1.4.7, mais il y en avait trop faire la liste	2016-06-16
1.4.7	Dernière version au 19 novembre 2016	2016-11-19
1.6.11		2017-05-25
1.7.3	Dernière version au 19 juin 2017	2017-06-19

Examples

Installation d'Electron

Les dépendances

Pour installer électron, vous devez d'abord installer Node.js , qui vient avec npm .

Comment l'installer?

Utilisez npm :

```
# Install the `electron` command globally in your $PATH
npm install electron -g

# OR

# Install as a development dependency
npm install electron --save-dev
```

Bonjour le monde!

Installer

Une structure de projet Electron ressemble généralement à ceci:

```
hello-world-app/
├── package.json
├── index.js
└── index.html
```

Maintenant, créons les fichiers et initialisons notre `package.json` .

```
$ mkdir hello-world-app && cd hello-world-app
$ touch index.js
$ touch index.html
$ npm init
```

Remarque: Si le paramètre `main` n'est pas spécifié dans `package.json` , Electron utilisera `index.js` comme point d'entrée par défaut.

Le processus principal

Dans Electron, le processus qui exécute le script principal de `package.json` s'appelle le **processus principal** . Ici, nous pouvons afficher une interface graphique en créant des instances de `BrowserWindow` .

Ajoutez ce qui suit à `index.js` :

```
const { app, BrowserWindow } = require('electron')

// Global reference to the window object
let win

// This method will be called when Electron has finished
// initialization and is ready to create browser windows
app.on('ready', function(){
    // Create the window
    win = new BrowserWindow({width: 800, height: 600})

    // Open and load index.html to the window
    win.loadURL('file://' + __dirname + '/index.html')

    // Emitted when the window is closed.
    win.on('closed', () => {
        // Dereference the window object
        win = null
    });
})

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

Modèle HTML et processus de rendu

Ensuite, nous créons l'interface graphique pour l'application. Electron utilise des pages Web comme interface graphique, chacune exécutant son propre processus appelé **processus de rendu** .

Ajoutez le code suivant à `index.html` :

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World</title>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
```

Lancer l'application

Il existe plusieurs façons d’exécuter une application Electron.

Avec `electron-prebuilt` installé globalement

Tout d’abord, assurez-vous d’avoir installé une `electron-prebuilt` installation d’ `electron-prebuilt` .

Maintenant, nous pouvons tester l'application en utilisant cette commande:

```
$ electron .
```

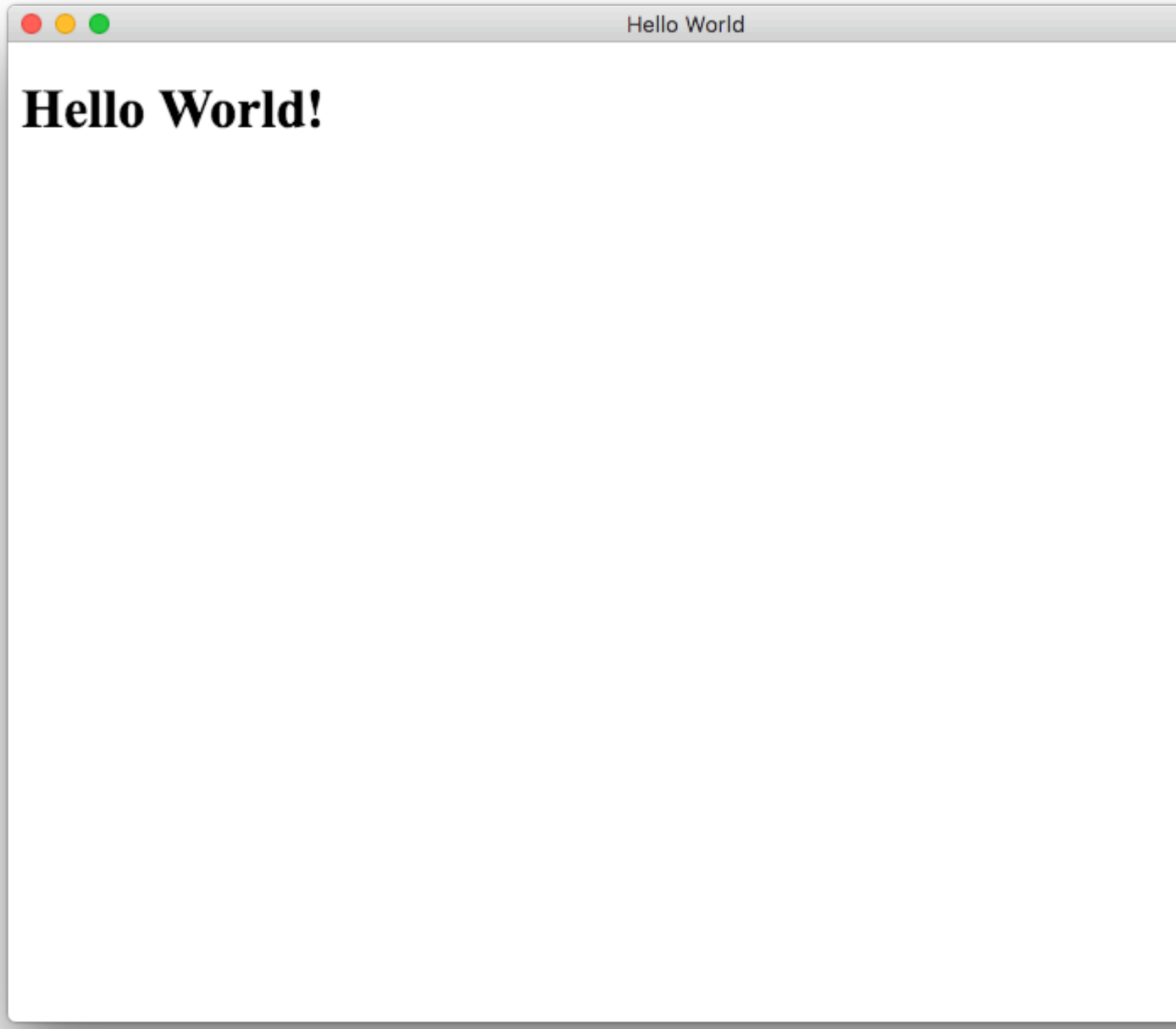
Méthode 2 - Sans `electron-prebuilt` installé globalement

Tout d'abord, nous devrons entrer le dossier de votre application (le dossier où est package.json).

Là, ouvrez une fenêtre d'invite de terminal / commande et tapez `npm install` pour installer le nécessaire dans le dossier de cette application.

Ensuite, `npm start` pour lancer l'application. N'oubliez pas que votre `package.json` doit toujours spécifier un script de démarrage.

Si tout fonctionnait correctement, vous devriez voir quelque chose comme ceci:



Toutes nos félicitations! Vous avez créé avec succès votre première application Electron.

Lire Commencer avec l'électron en ligne: https://riptutorial.com/fr/electron/topic/4934/commencer-avec-l-electron

Chapitre 2: App plateau électronique

Examples

App plateau électronique

Ajouter une icône à votre barre de tâches

```
let tray = null;
let mainWindow = null;
let user = null;

app.on('ready', () => {
    /**
     * Tray related code.
     */
    const iconName = 'icon.png';
    const iconPath = path.join(__dirname, iconName);
    tray = new Tray(iconPath);
    tray.setToolTip('AMP Notifier App');
    const contextMenu = Menu.buildFromTemplate([{
        label: 'Quit',
        click: destroyApp
    }]);
    tray.setContextMenu(contextMenu);

    tray.on('click', () => {
        app.quit();
    });
});
```

Lire App plateau électronique en ligne: https://riptutorial.com/fr/electron/topic/8160/app-plateau-electronique

Chapitre 3: électron-winstaller

Introduction

Module NPM qui crée des programmes d'installation Windows pour les applications Electron. Cela aidera à créer un EXE unique pour l'application Windows Electron

Syntaxe

- **Installer globalement**
- npm install -g electron-winstaller
- **Installer localement**
- npm install --save-dev électron-winstaller

Paramètres

Nom de configuration	La description
appDirectory	La valeur auteurs pour les métadonnées du package nuget. Par défaut, le champ auteur du fichier package.json de votre application n'est pas spécifié.
les propriétaires	La valeur du propriétaire pour les métadonnées du package nuget. Par défaut, le champ auteurs n'est pas spécifié.
EXE	Le nom du fichier principal de votre application .exe. Cela utilise le champ de nom dans le fichier package.json de votre application avec une extension .exe ajoutée si non spécifié.
la description	La valeur de description pour les métadonnées du package nuget. Par défaut, le champ de description du fichier package.json de votre application n'est pas spécifié.
version	La valeur de version pour les métadonnées du package nuget. Par défaut, le champ de version du fichier package.json de votre application n'est pas spécifié.
Titre	La valeur du titre pour les métadonnées du package nuget. Par défaut, le champ productName, puis le champ name du fichier package.json de votre application, si non spécifié.
prénom	ID de modèle d'application Windows (appId). Par défaut, le champ Nom du fichier package.json de votre application.

Nom de configuration	La description
certificateFile	Le chemin d'accès à un certificat de signature de code Authenticode
certificatePassword	Le mot de passe pour déchiffrer le certificat donné dans certificateFile
SignWithParams	Params à passer à signtool. Substitue certificateFile et certificatePassword.
iconUrl	Une URL vers un fichier ICO à utiliser comme icône de l'application (affichée dans Panneau de configuration> Programmes et fonctionnalités). Par défaut, l'icône Atom.
setupIcon	Le fichier ICO à utiliser comme icône pour le fichier Setup.exe généré
setupExe	Le nom à utiliser pour le fichier Setup.exe généré
setupMsi	Le nom à utiliser pour le fichier Setup.msi généré
noMsi	Squirrel.Windows devrait-il créer un programme d'installation MSI?
distanceReleases	Une URL vers vos mises à jour existantes. S'ils sont donnés, ils seront téléchargés pour créer des mises à jour delta
remoteToken	Jeton d'authentification pour les mises à jour à distance

Examples

Construire JS

Voici le fichier de construction de base pour construire un exécutable à partir de l'application windows électron.

```
var electronInstaller = require('electron-winstaller');
var resultPromise = electronInstaller.createWindowsInstaller({
    appDirectory: 'Your_electron_application_path',
    authors: 'Author Name',
    description: "Description"
});

resultPromise.then(() => console.log("Build Success!"), (e) => console.log(`No dice:
${e.message}`));
```

Lire électron-winstaller en ligne: https://riptutorial.com/fr/electron/topic/9492/electron-winstaller

Chapitre 4: Emballage d'une application électronique

Introduction

Lorsque vous êtes prêt pour la distribution, votre application électronique peut être intégrée dans un fichier exécutable.

Les applications Electron peuvent être packagées pour fonctionner sous Windows (32/64 bits), OSX (macOS) et Linux (x86 / x86_64).

Pour empaqueter votre code, utilisez le paquet npm 'electron-packager \

https://github.com/electron-userland/electron-packager

Syntaxe

- $ emballeur d'électrons
- sourcedir
- nom de l'application
- --platform = plate-forme
- --arch = arch
- [drapeaux facultatifs ...]

Paramètres

Paramètre	Détails
sourcedir	Le répertoire de vos fichiers d'application électronique
nom de l'application	Le nom de votre application
Plate-forme	La plate-forme pour laquelle vous souhaitez compiler votre code. Omettre cela compilera pour le système d'exploitation hôte
cambre	L'architecture du système pour laquelle vous souhaitez compiler votre code. Omettre cela compilera pour l'arc hôte

Examples

Installation du conditionneur d'électrons

```
# for use in npm scripts
npm install electron-packager --save-dev

# for use from cli
npm install electron-packager -g
```

Emballage de CLI

```
electron-packager C:/my-app MyApp
```

Emballage de script

```
var packager = require('electron-packager');

packager({
    dir: '/',
}, function(err, path){
    if(err) throw err;
    // Application has been packaged
});
```

Création de scripts npm pour automatiser le conditionnement Electron

Un moyen pratique de conditionner votre application consiste à écrire les scripts dans votre fichier `packages.json` et à les exécuter avec la commande `npm run`

```
{
    "name": "AppName",
    "productName": "AppName",
    "version": "0.1.1",
    "main": "main.js",
    "devDependencies": {
        "electron": "^1.6.6",
        "electron-packager": "^8.7.0"
    },
    "scripts": {
        "package-mac": "electron-packager . --overwrite --platform=darwin --arch=x64 --
icon=images/icon.png --prune=true --out=release-builds",
        "package-win": "electron-packager . --overwrite --platform=win32 --arch=ia32 --
icon=images/icon.png --prune=true --out=release-builds",
        "package-linux" : "electron-packager . --overwrite --platform=linux --arch=x64 --
icon=images/icon.png --prune=true --out=release-builds"
    }
}
```

Et pour les exécuter, il suffit d'écrire:

```
npm run package-mac
npm run package-win
npm run package-linux
```

Une ventilation des indicateurs de commande est la suivante:

```
electron-packager .     // this runs the packager in the current folder
--overwrite             // overwrite any previous build
--platform=darwin       // platform for which the binaries should be created
--arch=x64              // the OS architecture
--icon=images/icon.png  // the icon for the app executable
--prune=true            // this does not copy your dev-dependencies that appear in your
packages.json
--out=release-builds    // the name of the folder were the binaries will be outputed
```

Auparavant, l'exécution des scripts modifiait les dépendances de devDependons, car électron-packager ne peut pas regrouper les packages des dépendances devDepend dans l'application. Dans packager.json, changez le mot (s'il y en a ou si les paquets sont installés en utilisant --save-dev dans npm install) devDependencies aux seules dépendances.

Lire Emballage d'une application électronique en ligne:
https://riptutorial.com/fr/electron/topic/8945/emballage-d-une-application-electronique

Chapitre 5: Fonction à distance - utilisez les fonctions électroniques en JavaScript

Introduction

Si vous devez changer certaines choses dans renderer.js ou main.js mais que vous voulez faire les modifications dans index.html, vous pouvez utiliser la fonction distante. Il vous permet d'accéder à toutes les fonctions électroniques dont vous avez besoin!

Syntaxe

- utiliser remote comme `require("electron")` :
    - main.js: `const electron = require("electron");`

      index.html: `const electron = require("electron").remote;`

Examples

Utilisation de la télécommande en définissant la barre de progression

```
const { remote } = require("electron"); // <- The Node.js require() function is
                                        // added to JavaScript by electron

function setProgress(p) { // p = number from 0 to 1
  const currentWindow = remote.getCurrentWindow();
  currentWindow.setProgressBar(p);
}
```

Utilisation de la télécommande en définissant la fenêtre en plein écran

```
const { remote } = require("electron"); // <- The Node.js require() function is
                                        // added to JavaScript by electron

function fullscreen(f) { // p = false or true
  const currentWindow = remote.getCurrentWindow();
  currentWindow.maximize();
}
```

Lire Fonction à distance - utilisez les fonctions électroniques en JavaScript en ligne: https://riptutorial.com/fr/electron/topic/8719/fonction-a-distance---utilisez-les-fonctions-electroniques-en-javascript

Chapitre 6: Processus principal et rendu.

Remarques

Le processus qui exécute le script principal de `package.json` est appelé **processus principal** . Le processus principal crée des pages Web en créant des instances de `BrowserWindow` . Chaque page Web dans Electron s'exécute dans son propre processus, appelé **processus de rendu** . Le processus principal gère toutes les pages Web et leurs processus de rendu correspondants. Chaque processus de rendu est isolé et ne s'intéresse qu'à la page Web exécutée.

Examples

Communication IPC asynchrone

Code source du processus principal `index.js` :

```
const {app, BrowserWindow, ipcMain} = require('electron')
let win = null

app.on('ready', () => {
  win = new BrowserWindow()
  win.loadURL(`file://${__dirname}/index.html`)
  win.webContents.openDevTools()
  win.on('closed', () => {
    win = null
  })
  win.webContents.on('did-finish-load', () => {
    win.webContents.send('asyncChannelToRenderer', 'hello')
  })
})

ipcMain.on('asyncChannelToMain', (event, arg) => {
  console.log(arg + ' from renderer')
  if (arg === 'hello') {
    event.sender.send('asyncChannelToRenderer', 'world')
  }
})
```

Processus de `index.html` dans `index.html` :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World IPC</title>
    <script>
      require('electron').ipcRenderer.on('asyncChannelToRenderer', (event, arg) => {
        console.log(arg + ' from main')
        if (arg === 'hello') {
          event.sender.send('asyncChannelToMain', 'world')
        }
      })
    </script>
```

```
  </head>
  <body>
    <button onclick="require('electron').ipcRenderer.send('asyncChannelToMain',
'hello')">click me</button>
  </body>
</html>
```

Module RMI à distance

Le module `remote` permet une invocation RMI (Remote Method Invocation) simple des objets de processus principaux du processus de rendu. Commencez par créer le processus principal dans `index.js`

```
const {app, BrowserWindow} = require('electron')
let win = null

app.on('ready', () => {
  win = new BrowserWindow()
  win.loadURL(`file://${__dirname}/index.html`)
  win.on('closed', () => {
    win = null
  })
})
```

puis le processus distant `index.html`

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      const {BrowserWindow, app} = require('electron').remote
    </script>
  </head>
  <body>
    <button onclick= "let win = new BrowserWindow();
win.loadURL(`file://${__dirname}/index.html`)">new window</button>
    <button onclick= "app.quit()">quit</button>
  </body>
</html>
```

Communication IPC synchrone

Créer `index.js` comme

```
const {app, BrowserWindow, ipcMain} = require('electron')
let win = null

app.on('ready', () => {
  win = new BrowserWindow()
  win.loadURL(`file://${__dirname}/index.html`)
  win.webContents.openDevTools()
  win.on('closed', () => {
    win = null
  })
})
```

```
ipcMain.on('syncChannelToMain', (event, arg) => {
  console.log(arg + ' from renderer')
  event.returnValue = 'world'
})
```

et le moteur de rendu `index.html` as

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World IPC</title>
  </head>
  <body>
    <button onclick="console.log(require('electron').ipcRenderer.sendSync('syncChannelToMain',
'world') + ' from main')">click me</button>
  </body>
</html>
```

Lire Processus principal et rendu. en ligne: https://riptutorial.com/fr/electron/topic/5432/processus-principal-et-rendu-

Chapitre 7: Utiliser le bootstrap en électron

Introduction

L'un des meilleurs frameworks frontaux dans le monde du web en twitter bootstrap. Comme l'électron s'appuie sur un navigateur Web, nous pouvons facilement utiliser le bootstrap avec électron pour utiliser la puissance du bootstrap dans notre structure électronique. La dernière version de bootstrap à ce jour est 3.3.7 et bootstrap 4 est toujours en phase alpha.

Examples

Relier l'électron au bootstrap

Pour utiliser bootstrap, il y a 2 cas.

1. L'application électronique est connectée à internet
2. L'application électronique n'est pas connectée à Internet

Pour les applications électroniques connectées à Internet, nous pouvons simplement utiliser des liens CDN pour bootstrap et les inclure dans nos fichiers HTML.

Le problème vient quand nous devons le prendre en version hors connexion où l'application n'est pas connectée au réseau. Dans ce cas,

1. Télécharger le bootstrap de Bootstrap
2. Décompressez le dossier dans l'application électronique
3. Dans le répertoire bootstrap, il y a des fichiers css et javascript.
4. Pour une meilleure compréhension, déplacez les fichiers css bootstrap dans le dossier CSS (tous les fichiers de style seront dans ce dossier) et les fichiers jst bootstrap dans le dossier JS (tous les fichiers Javascript seront dans ce dossier)
5. Dans vos fichiers HTML, liez les fichiers HTML en utilisant le code suivant

```
<link rel="stylesheet" href="path_to_the_offline_bootstrap_css_file">
<script scr="path_to_the_offline_bootstrap_js_file"></script>
```

De cette façon, vous pouvez commencer à utiliser twitter bootstrap dans un framework électronique.

Lire Utiliser le bootstrap en électron en ligne: https://riptutorial.com/fr/electron/topic/10897/utiliser-le-bootstrap-en-electron

Chapitre 8: Utiliser nedb en électron

Examples

Installation de nedb

Il est très facile d'installer nedb.

```
npm install nedb --save    # Put latest version in your package.json
```

Pour les gens aimants,

```
bower install nedb
```

Application électronique de connexion avec Nedb

Lors de la création d'applications électroniques, le backend se trouve généralement dans un dossier séparé (fichiers js) et le frontal se trouve dans un dossier séparé (fichiers html). Dans le backend, pour utiliser la base de données, nous devons inclure le package nedb avec la déclaration de nécessité comme suit.

```
var Datastore = require('nedb'),db = new Datastore({ filename: 'data.db', autoload: true });
```

Gardez à l'esprit que le chargement du fichier de base de données est une tâche asynchrone.

Insérer des données dans nedb

Fondamentalement, pour insérer des enregistrements dans nedb, les données sont stockées sous la forme de json avec la clé étant les noms de colonne et la valeur de ces noms sera les valeurs de cet enregistrement.

```
var rec = { name: 'bigbounty',age:16};

db.insert(rec, function (err, newrec) {   // Callback is optional
  // newrec is the newly inserted document, including its _id
  // newrec has no key called notToBeSaved since its value was undefined
});
```

Soyez prudent avec toutes les opérations de la base de données, car elles sont asynchrones.

Note **: Si _id n'est pas présent dans les données json que vous insérez automatiquement, il sera créé pour vous par nedb.

Rechercher dans nedb

Afin de rechercher des enregistrements dans nedb, nous devons à nouveau passer le json

contenant les critères de recherche en tant que paramètre à la fonction find de l'objet db.

```
db.find({ name: 'bigbounty' }, function (err, docs) {
  // docs is an array containing documents that have name as bigbounty
  // If no document is found, docs is equal to []
});
```

Pour ne trouver qu'un seul document, comme dans l'utilisation de limit dans mysql, c'est facile dans nedb.

```
db.findOne({ name: 'bigbounty' }, function (err, doc) {
  // doc is only one document that has name as bigbounty
  // If no document is found, docs is equal to []
});
```

Supprimer dans nedb

Pour supprimer des documents dans nedb, c'est très simple. Nous devons simplement utiliser la fonction remove de l'objet db.

db.remove ({name: 'bigbounty'}, function (err, numremoved) {// numremoved est le nombre de documents supprimés.});

Lire Utiliser nedb en électron en ligne: https://riptutorial.com/fr/electron/topic/10906/utiliser-nedb-en-electron

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec l'électron	Alphonsus, Community, Eslam Mahmoud, Florian Hämmerle, Hewbot, J F, Piero Divasto, SimplyCodin, Theo, vintproykt, Vishal
2	App plateau électronique	Anavar Bharmal, nmnsud
3	électron-winstaller	Krupesh Kotecha
4	Emballage d'une application électronique	bigbounty, Dan Johnson, VladNeacsu
5	Fonction à distance - utilisez les fonctions électroniques en JavaScript	B. Colin Tim, Florian Hämmerle
6	Processus principal et rendu.	mrkovec
7	Utiliser le bootstrap en électron	bigbounty
8	Utiliser nedb en électron	bigbounty