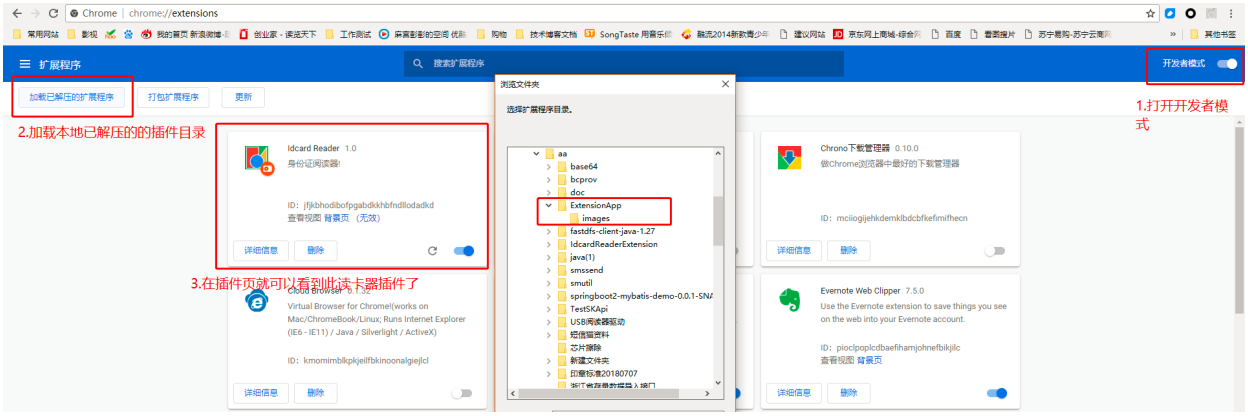
一：谷歌插件和外接硬件设备交互步骤和原理

注：这里只做一个demo 通过网页调用插件来连接exe与硬件交互

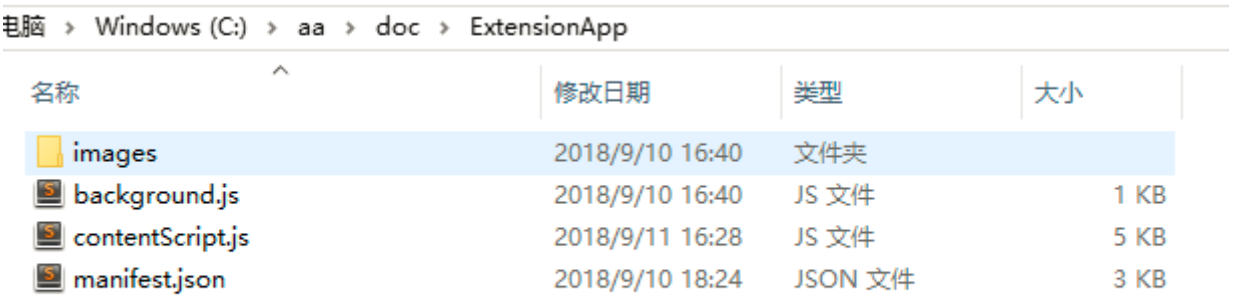
安装步骤：

1.环境安装：

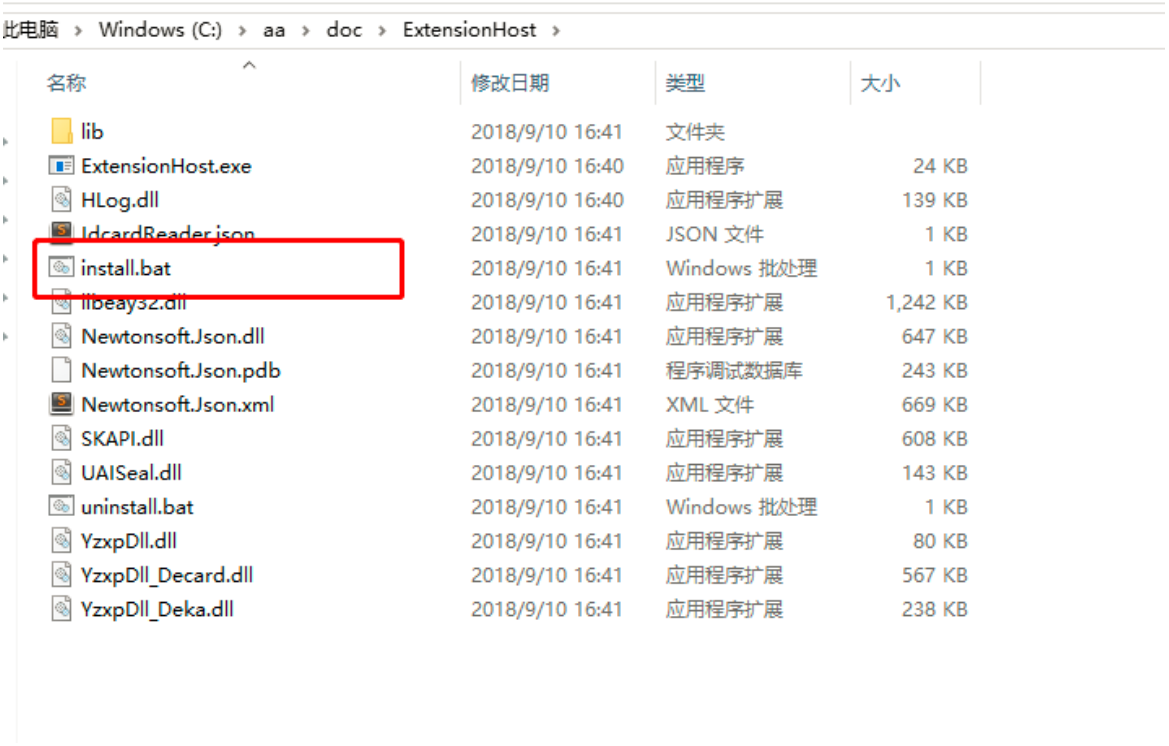
1.1 chrome浏览器打包插件程序



插件目录的文件：



1.2 安装插件和exe交互的host环境，点击install.bat即安装完成



1.3安装读卡器或外接硬件设备驱动（身份证插件已绝大多数读卡器）

致此环境已安装完成；可试运行在页面看效果；

2.网页数据，谷歌插件，可执行文件exe，硬件设备交互流程和原理

2.1.页面的数据只会跟插件交互；安装插件后，可将网页数据通过js事件发送给插件；

2.1.1：页面加载时同时检测插件并监听插件

```
<body onload="checkExtension()">
    <img id="photo" style="width:90px;height:120px;" src=""/>
    <button id='connect-button' onClick="sealVerify();">读取芯片</button>
    <button id='connect-button' onClick="readCard();">身份证读卡</button>
    <button id='connect-button' onClick="sealRevoke();">芯片注销</button>
    <button id='test-button' onClick="personal();">个人化</button>

    <div id='response'></div>
</body>
```

2.1.2：点击身份证读卡时，向插件发送数据或者指令；

```
function readCard() {
  if (!extensionNode) {
      alert('读卡器插件未安装！');
      return;
  }
  var readCardEvent = document.createEvent('Event');
  readCardEvent.initEvent('ReadIDCard', true, true);
  // 发出事件
  extensionNode.dispatchEvent(readCardEvent);
}
```

2.1.3：content.js监听到网页的发出事件并向background.js发送指令（json格式）要开启长连接：

```
port = chrome.runtime.connect();
```

```
}

//注册身份证事件监听
function registReadCardEvent() {
    extensionNode.addEventListener('ReadIDCard', function (evt) {
        port.postMessage({"command": Command_IDCARD_READ});
    });
}
```

2.1.4：background.js将指令数据发送给可执行文件exe；通过（native message以json数据格式）数据的大小要控制；图片可转换为base64字符串；

contentScript.js | background.js

```
'use strict';

{
    var hostName = "dhht.idcard.reader";
    var nativePort;
    chrome.runtime.onConnect.addListener(function(port) {
        port.onMessage.addListener(function(command) {
            if(nativePort == null) {
                console.log("connect");
                nativePort = chrome.runtime.connectNative(hostName);
                nativePort.onMessage.addListener(function(response) {
                    console.log(response);
                    port.postMessage(response);
                });
                nativePort.onDisconnect.addListener(function() {
                    nativePort = null;
                    console.log("host exit!");
                });
            }
            console.log(command);
            nativePort.postMessage(command);
        });
    });
}

```

2.1.5：exe：处理完成之后也会返回json字符串；然后取得里面的数据输出到页面上

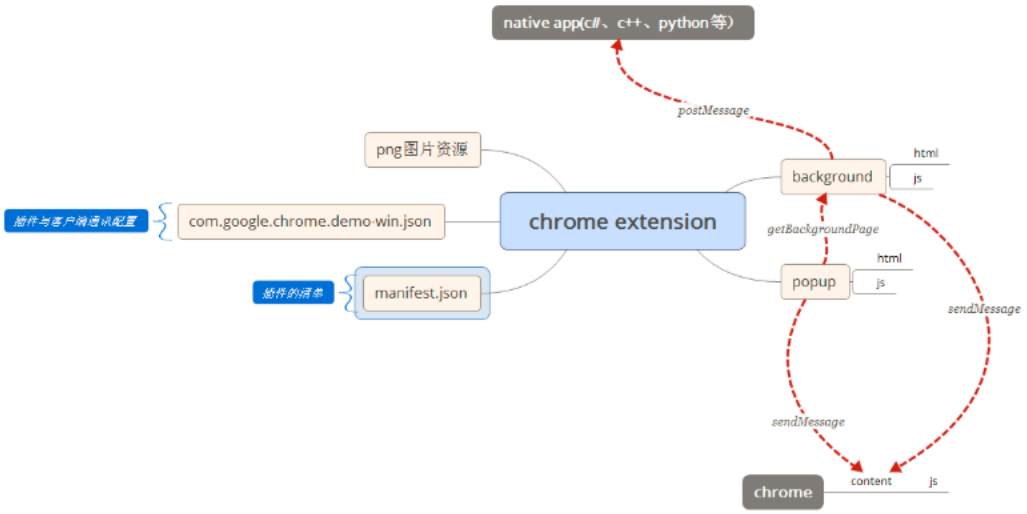
```
// port.postMessage({joke: "Knock knock"});
port.onMessage.addListener(function(msg) {
    console.log(msg);
    // if (msg.question == "Who's there?")
    //     port.postMessage({answer: "Madame"});
    // else if (msg.question == "Madame who?")
    //     port.postMessage({answer: "Madame... Bovary"});
    if(msg.command == 0) {
        return;
    }
    var eventName = "";
    switch (msg.command)  {
        case Command_IDCARD_READ : eventName = "IDCardReadResult";break;
        case Command_SEAL_PERSONAL : eventName = "PersonalResult";break;
        case Command_SEAL_VERIFY :  eventName = "VerifyResult";break;
        case Command_SEAL_PROC_PERCENT : eventName = "SealProcessPercent";break;
        case Command_SEAL_PERSONAL_DATA : eventName = "PersonalData";break;
        case Command_SEAL_DATA_REVOKE : eventName = "RevokeResult";break;
        case Command_SEAL_REVOKE_DATA : eventName = "RevokeData";break;

    }

    extensionResponseNode.innerText = JSON.stringify(msg);
    var readCardResultEvent = document.createEvent('Event');
    readCardResultEvent.initEvent(eventName, true, true);
    // 发出事件
    extensionNode.dispatchEvent(readCardResultEvent);
});
```

交互流程如下图：

这就是我在基础篇中提出的三种js的通讯，我以一张图概括之：



源码地址：https://github.com/yichen520/IDcardExtensionApp

注意：这是写自己匹配的地址：测试时需要一个服务器环境；

```
{
  "key":                                                    SkAgEAAoIBAQDSVwCH8Efc
  "name": "Idcard Reader",
  "version": "1.0",
  "description": "身份证阅读器!",
  "permissions": [
    "declarativeContent",
    "nativeMessaging"
  ],
  "background": {
    "scripts": [
      "background.js"
    ],
    "persistent": false
  },
  "externally_connectable": {
    "matches": ["*://test.example.com/*"]
  },
  "content_scripts": [
    {
      "matches": [
        "*://*/test.html",
        "*://*/test"
      ],
      "js" : [
        "contentScript.js"
      ]
    }
  ],
  "icons": {
    "16": "images/icon16.png",
    "32": "images/icon32.png",
    "48": "images/icon48.png",
    "128": "images/icon128.png"
  },
  "manifest_version": 2
}
```