# 二、内容脚本

## 零、概念：

1. 内容脚本是在网页环境中运行的文件
2. 运行用于读取和修改网页内容的脚本。
3. 生命周期跟随页面
4. 有跨域的问题

```
"content_scripts": [
  {
    "js": ["scripts/content.js"],
    "matches": [//这些变量可让浏览器识别要将内容脚本注入到哪些网站，至少要写一个匹配项
      "https://developer.chrome.com/docs/extensions/*",
      "https://developer.chrome.com/docs/webstore/*"
    ]
  },
```

## 一、只能访问以下api，其他的不可以：

- dom
- i18n
- storage
- runtime.connect()
- runtime.getManifest()
- runtime.getURL()
- runtime.id
- runtime.onConnect
- runtime.onMessage
- runtime.sendMessage()

## 二、声明方式：

1. **注入静态声明：在mainfest.json里添加**

```
"content_scripts": [
  {
    "matches": ["https://*.nytimes.com/*”],//匹配的地址
    "css": ["my-styles.css”],//注入的css
    "js": ["content-script.js”]//注入的js
    “run_at”:“document_idle”//首选，document_start:任何css文件之后，document_end：dom完成后资源加载前注入。
  }
],
```

2. **注入动态声明**

内容脚本对象是使用 chrome.scripting 中的方法在 Chrome 中注册的。
chrome.scripting.registerContentScripts 注册
. updateContentScripts 更新

. getRegisteredContentScripts 获取注册内容脚本
. unregisterContentScripts 取消注册

3. **编程方式注入**

需要开放主机权限：

```
"permissions": [
   "activeTab",
   "scripting"
 ],
```

点击 tab 时触发

```
function injectedFunction() {
 document.body.style.backgroundColor = "orange";
}

chrome.action.onClicked.addListener((tab) => {
 chrome.scripting.executeScript({
   target: { tabId: tab.id },
   files: ["content-script.js"] 插入 js 文件
   func：injectedFunction 也可以注入函数，但是函数的变量只能是内部的
 });
});
```

## 三、match 匹配模式

1. https://*/* 或 https://*/：　匹配使用 https 架构的所有网址
2. "<all_urls>”：所有网址
3. "file:///“：本地文件上运行
4. http://localhost/*：开发期间匹配任何 localhost 端口
5. **不支持顶级域名**：.com 、.net 等