IntersectionObserver API 使用教程

::: tip 用途 网页开发时，常常需要了解某个元素是否进入了"视口"（viewport），即用户能不能看到它。 :::

简单使用

```
var io = new IntersectionObserver(callback, option);
// 开始观察
io.observe(document.getElementById("example"));

// 停止观察
```

```
io.unobserve(element);

// 关闭观察器
io.disconnect();
```

查看详情 IntersectionObserver

GIT git 流水线技巧

```
我代码已经恢复了 git reflog  然后git reset --hard HEAD{x}  x填你要回滚的那一步就ok了我
代码已经恢复了 git reflog  然后git reset --hard HEAD{x}  x填你要回滚的那一步就ok了
git reflog  然后git reset --hard HEAD{x}  x填你要回滚的那一步就ok了
git stash    (将自己分支修改的内容暂存起来)
git checkout 分支名  (一般切换到主分支)
git pull origin 分支名   (一般拉取最新的主分支代码)
git checkout 自己分支名     (再切换到自己分支)
git merge 主分支名       (将主分支合并到自己分支，有冲突解决冲突)
git stash pop    (将自己改的暂存的内容放出来,有冲突解决冲突)
git add .    (缓存代码)
git commit -m'修改说明'    (修改记录日志)
git push origin 自己分支名   (将自己分支提交到git仓库)
git merge –abort 可以停止合并
git log 查看历史提交

git branch 创建新分支
没有参数时 git branch 会列出你在本地的分支
我们也可以使用 git checkout -b (branchname) 命令来创建新分支并立即切换到该分支下，从而在
该分支中操作。

删除分支命令:
git branch -d (branchname)

cd old-repo (进入旧的仓库目录)
git remote set-url origin ssh://git@bitbucket.sw.nxp.com/dash/nfc-module.git  (设定
新的远程仓库地址)
git push -u origin --all (将所有的分支都推送到新的仓库)
git push origin --tags  (将所有tag都推送到新的仓库)

git merge --abort  git取消合并
```

git 如何退出 VIM 编辑器

按下 esc

接着 q! 回车 :表示不保存退出

或 wq! 回车 表示保存并退出

git log 按 q 退出

- :q ，退出（:quit 的缩写）
- :q!，退出且不保存（:quit!的缩写）
- :wq，保存并退出
- :wq!，保存并退出即使文件没有写入权限（强制保存退出）
- :x，保存并退出（类似:wq，但是只有在有更改的情况下才保存）
- :exit，保存并退出（和:x 相同）
- :qa，退出所有(:quitall 的缩写)
- :cq，退出且不保存（即便有错误）

github 创建个人访问令牌

::: tip 介绍 github 已不再支持密码做验证登录 需要自己创建个人访问令牌做权限管理和分配 ::: ::: warning 注意 令牌一定要记得复制保存 处于安全考虑 当你退出当前令牌生成页面之后你将看不到令牌代码 :::

1. 点击个人中心头像 选择 Developer settings（开发人员设置）
2. 选择 Personal access tokens （个人访问令牌）
3. 点击 Generate new tokne （设置新的访问令牌）
4. 选择权限分配完成之后点击 Generate tokne（生成令牌）
5. 拿到令牌 复制保存
6. 使用令牌 正常连接 github 会给你调起登录框 密码框中不在输入密码输入令牌即可

ES6-N

::: tip 介绍 介绍 ES6-ES13 中的新增特性 :::

ECMAScript 2022(ES13)提案阶段，更倾向于面向对象

1. 声明类的字段：类字段可以在类的顶层被定义和初始化
2. 私有方法&字段：用#前缀来定义类的私有方法和字段
3. 类的静态公共方法和字段：增加了静态公共字段、静态私有方法和静态私有字段的特性
4. ECMScript 类静态初始化块：在类声明/定义期间评估静态初始化代码块，可以访问类的私有字段

5. 检测私有字段：可以使用 in 操作符，如果指定的属性/字段在指定的对象/类中，则返回真，并且也能判断私有字段
6. 正则匹配索引：该提案提供了一个新的/dflag，以获得关于输入字符串中每个匹配的开始和索引位置结束的额外信息
7. 在所有内置的可索引数据上新增.at()方法
8. Object.hasOwn(object, property)：使用 Object.hasOwn 替代 Object.prototype.hasOwnProperty.call
9. Error Cause：为了便捷的传递导致错误的原因

ECMAScript 2021 (ES12)

1. String.prototype.replaceAll ：有了这个 API，替换字符不用写正则了
2. Promise.any() ：返回第一个 fullfilled 的 promise ，若全部 reject，则返回一个带有失败原因的 AggregateError。
3. 新增逻辑赋值操作符： ??=、&&=、 ||=
4. WeakRefs：使用弱引用对象，该弱引用不会阻止 GC，并且可以在 GC 前使用 WeakRef.prototype.deref ( ) 解除该引用。
5. 下划线 (_) 分隔符：使用 _ 分隔数字字面量以方便阅读
6. Intl.ListFormat ：用来处理和多语言相关的对象格式化操作
7. Intl.DateTimeFormat API 中的 dateStyle 和 timeStyle 的配置项：用来处理多语言下的时间日期格式化的函数

ECMAScript 2019(ES11)

1. 动态 import ()：按需导入
2. 空值合并运算符：表达式在 ?? 的左侧 运算符求值为 undefined 或 null，返回其右侧
3. 可选链接：?.用户检测不确定的中间节点
4. BigInt：新基本数据类型，表示任意精度的整数
5. globalThis：浏览器：window、worker：self、node：global
6. Promise.allSettled：返回一个在所有给定的 promise 已被决议或被拒绝后决议的 promise，并带有一个对象数组，每个对象表示对应的 promise 结果
7. for-in 结构：用于规范 for-in 语句的遍历顺序

ECMAScript 2019(ES10)

1. Array.flat()和 Array.flatMap()：数组展平
2. String.trimStart()和 String.trimEnd()：去掉开头结尾空格文本
3. String.prototype.matchAll：为所有匹配的匹配对象返回一个迭代器

4. Symbol.prototype.description：只读属性，回 Symbol 对象的可选描述的字符串
5. Object.fromEntries()：返回一个给定对象自身可枚举属性的键值对数组
6. 可选 Catch
7. SON Superset 超集
8. JSON.stringify() 加强格式转化
9. Array.prototype.sort() 更加稳定
10. Function.prototype.toString() 重新修订

ECMAScript 2018(ES9)

1. 异步迭代：await 可以和 for...of 循环一起使用，以串行的方式运行异步操作
2. Promise.finally()：逻辑只可以放在一个地方，这有点像以前 jQuery ajax 的 complete
3. Rest/Spread 属性：允许我们将一个剩余参数表示为一个数组正则表达式命名捕获组：允许命名捕获组使用符号?< name >
4. 正则表达式反向断言(lookbehind)
5. 正则表达式 dotAll 模式：正则表达式中点.匹配除回车外的任何单字符，标记 s 改变这种行为，允许行终止符的出现
6. 正则表达式 Unicode 转义: Unicode 属性转义形式为\p{...}和\P{...}

ECMAScript 2017(ES8)

1. async/await: 异步终极解决方案
2. Object.values()
3. Object.entries()
4. String padding：String.prototype.padStart、String.prototype.padEnd
5. 函数参数列表结尾允许逗号
6. Object.getOwnPropertyDescriptors(): 获取一个对象的所有自身属性的描述符,如果没有任何自身属性，则返回空对象
7. SharedArrayBuffer 对象：用来表示一个通用的，固定长度的原始二进制数据缓冲区
8. Atomics 对象：提供了一组静态方法用来对 SharedArrayBuffer 对象进行原子操作

ECMAScript 2016(ES7)

1. Array.prototype.includes()
2. 指数操作符 **

ECMAScript 2015(ES6)

1. let 和 const
2. 类（class）
3. 模块化(ES Module)
4. 箭头（Arrow）函数
5. 函数参数默认值
6. 模板字符串
7. 解构赋值
8. 延展操作符 ...
9. 对象属性简写
10. Promise

查看详情 原作者地址

NoPrint.js--JavaScript 在 HTML 中禁用打印、截屏、复制和粘贴

::: tip 用途 禁用复制和粘贴 禁用打印 禁用鼠标右键 禁用截图 禁用“另存为”/ Ctrl + S 自动模糊 :::

简单使用

```
//引入noprint.js cdn地址 https://pdfanticopy.com/noprint.js
var noPrint = true; //noPrint将禁用 CTRL + P 快捷键和打印功能。即使浏览者尝试通过浏览器菜单或按钮打印网页，打印时内容也会变成空白页面。可以将noPrint设置为false以关闭打印保护。
var noCopy = true; //noCopy将禁用文本选择、鼠标右键、CTRL + S（保存网页）以及复制和粘贴功能。可以将noCopy设置为false以关闭此保护。
var noScreenshot = true; //noScreenshot将禁用PrintScreen键以防止访问者轻松截屏和获取网页快照。可以将 noScreenshot设置为false以关闭此功能。
var autoBlur = true; //一旦鼠标光标离开内容区域，autoBlur 会将网页上的所有内容变为模糊。浏览者可以通过单击网页来让内容变回清晰的样子。这可以防止人们通过第三方应用程序或 Opera工具栏上的快照功能截取屏幕截图。可以将autoBlur设置为false以禁用自动模糊
```

官网地址 noprint.js

js-cookie 轻量级的处理 cookies 的 js API

::: tip 用途 js-cookie 是一个简单的，轻量级的处理 cookies 的 js API。 :::

相关介绍 js-cookie

react-dnd 拖拽库

::: tip 用途 React DnD 是 React 和 Redux 的核心作者 Dan Abramov 创造的一组 React 高阶组件，可以在保持组件分离的前提下帮助构建复杂的拖放接口。 :::

相关介绍 react-dnd github react-dnd

Velocity.js

::: tip 用途 Velocity 是一个简单易用、高性能、功能丰富的轻量级 JS 动画库。它能和 jQuery 完美协作，并和$.animate()有相同的 API， 但它不依赖 jQuery，可单独使用。Velocity 不仅包含了 $.animate() 的全部功能， 还拥有：颜色动画、转换动画 (transforms)、循环、 缓动、SVG 动画、和 滚动动画 等特色功能。 :::

中文文档 Velocity.js

vue 中下载多个文件打包为一个 zip

需要使用到的库

```
cnpm install -S axios
cnpm install jszip
cnpm install file-saver
```

具体使用

```
<template>
  <div class="home">
     {{title}}
  </div>
</template>

<style lang="scss" scoped>
</style>
<script>
```

```
import JSZip from "jszip";
import FileSaver from "file-saver";
import axios from "axios";
const getFile = (url) => {
  return new Promise((resolve, reject) => {
    axios({
      method: "get",
      url,
      responseType: "arraybuffer",
    })
      .then((data) => {
        resolve(data.data);
      })
      .catch((error) => {
        reject(error.toString());
      });
  });
};
export default {
  name: "App",
  data(){
    return{
       title:""
    }
  },
  created() {
    this.filesToRar(
      [
        {
          fileUrl: "https://test.hamatd.com/static/test.pdf",
          renameFileName: "PDFtest.pdf",
        },
      ],
      "测试压缩包"
    );
  },
  watch: {},
  methods: {
    /**文件打包
     * arrImages:文件list:[{fileUrl:文件url,renameFileName:文件名}]
     * filename 压缩包名
     * */
    filesToRar(arrImages, filename) {
      let _this = this;
      let zip = new JSZip();
      let cache = {};
      let promises = [];
      _this.title = "正在加载压缩文件";
      for (let item of arrImages) {
        const promise = getFile(item.fileUrl).then((data) => {
          console.log(data)
          // 下载文件，并存成ArrayBuffer对象
          const file_name = item.renameFileName; // 获取文件名
          zip.file(file_name, data, { binary: true }); // 逐个添加文件
          cache[file_name] = data;
        });
        promises.push(promise);
      }
      Promise.all(promises)
```

```
        .then(() => {
          zip.generateAsync({ type: "blob" }).then((content) => {
            _this.title = "正在压缩";
            // 生成二进制流
            FileSaver.saveAs(content, filename); // 利用file-saver保存文件  自定义文
件名
            _this.title = "压缩完成";
          });
        })
        .catch((reserr) => {
            _this.title = '失败'+reserr;
          console.log('失败',reserr)
        });
    },
  },
};
</script>
```

各种文件类型文件类型

后缀名	文件类型	类型（**type**）
.xls	Microsoft Excel	application/vnd.ms-excel