index.js 使用指南

index.js 是一个功能丰富的 JavaScript 文件，提供了多个类和工具函数，用于处理浏览器环境中的 DOM 操作、事件处理、动画、文件加载等功能。以下是对该文件的详细使用指南。

1. 全局设置与初始化

1.1 全局变量与初始化

- globalThis.HTMLElement、globalThis.clearInterval、globalThis.setInterval：这些全局变量用于确保在非浏览器环境中也能正常运行。
- globalThis.performance：用于性能计时，如果浏览器不支持 performance，则使用 Date 作为替代。

1.2 浏览器环境检测

- browserMode：检测当前是否在浏览器环境中运行。如果 document 存在于 globalThis 中，则为 true。

2. 核心类

2.1 RealWorld 类

RealWorld 是一个事件循环类，用于管理异步任务的执行。

主要方法：

- **onload**：返回一个 Promise，当页面加载完成时兑现。
- **onceIf(ifFn)**：生成一个条件检测的 Promise，当 ifFn 返回 true 时兑现。
- **cb2promise**：将回调函数转换为 Promise。
- **destroy()**：销毁当前对象，清除定时器。
- **then(fn)**：将函数添加到执行队列中。

2.2 RealNode 类

RealNode 是一个基础类，用于管理节点及其相关操作。

主要方法：

- **createExpression(get, ...relativeRNs)**：创建一个表达式节点。
- **check(realNode)**：检查节点是否有效。
- **afterNow(fn, keepNow, thisArg, ...argArray)**：在当前事件循环后执行函数。
- **time(promise)**：测量 Promise 的执行时间。
- **copyObj(obj)**：深度复制对象。

2.3 RealElement 类

RealElement 继承自 RealNode，用于管理 HTML 元素。

主要方法：

- `getRealElement()`：生成一个 `RealElement` 实例。
- `applyCSS(selfSelector, classNameOrRuleObjObj)`：应用 CSS 样式。
- `clone(keepValue, fix, deepCopyRelativeRNs)`：克隆当前元素。
- `addClassName(...className)`：添加类名。
- `removeClassName(...className)`：移除类名。

2.4 `RealCanvas` 类

`RealCanvas` 继承自 `RealElement`，用于管理 HTML Canvas 元素。

主要方法：

- `getImageBySrc(src)`：通过 `src` 获取图像。
- `clear()`：清除画布内容。
- `toBlob()`：将画布内容转换为 `Blob`。
- `animate(config)`：执行动画。

2.5 `RealLoader` 类

`RealLoader` 继承自 `RealElement`，用于文件加载。

主要方法：

- `load(realLoader)`：加载文件。
- `fileName`：获取或设置文件名。

2.6 `RealSelect` 类

`RealSelect` 继承自 `RealElement`，用于管理 HTML `<select>` 元素。

主要方法：

- `protoTransform(value)`：将值转换为 HTML 选项。
- `fix()`：更新选项列表。

2.7 `RealComtag` 类

`RealComtag` 继承自 `RealElement`，用于管理复杂的 HTML 元素组合。

主要方法：

- `fix()`：更新元素内容。
- `protoTransform(value)`：将值转换为 HTML 元素。

2.8 `RealDivList` 类

`RealDivList` 继承自 `RealElement`，用于管理一组 `<div>` 元素。

主要方法：

- `createList(length, tagName, id, selfAssign)`：创建一组 `<div>` 元素。
- `defineDivListClass(className, tryHTML, optionList, tryRealNode, cssRuleObjObj, callback)`：定义一个新的 `RealDivList` 类。
- `createByClassName(className, ...argArray)`：通过类名创建 `RealDivList` 实例。

2.9 `RealImgList` 类

`RealImgList` 继承自 `RealDivList`，用于管理一组 `<img>` 元素。

主要方法：

- `cloneImgList()`：克隆图像列表。
- `protoTransform(value)`：将值转换为图像元素。

2.10 `RealDivQueue` 类

`RealDivQueue` 继承自 `RealDivList`，用于管理一组 `<div>` 元素的队列。

主要方法：

- `applyQueue(queueArray, target0, target1)`：应用队列顺序。
- `getListQueue()`：获取队列列表。

3. 工具函数

3.1 `createRealDivSelect(optionConfig, multiple, onchange)`

创建一个自定义的 `<div>` 选择器。

3.2 `createRealDivSearch(placeholder)`

创建一个自定义的搜索框。

3.3 `RealStory` 类

`RealStory` 用于管理页面和异步任务。

主要方法：

- `newPage()`：创建一个新页面。
- `then(fn)`：将函数添加到执行队列中。
- `clear()`：清除当前页面的所有任务。

3.4 `RealPromise`

`RealPromise` 是一个自定义的 `Promise` 实现，用于管理异步任务。

4. 示例用法

4.1 使用 `RealCanvas` 绘制图像

```
const canvas = new RealCanvas('myCanvas', 800, 600);
canvas.animate({
    prefix: './img/frame_',
    suffix: '.png',
    startN: 1,
    length: 100,
    timeSep: 100
});
```

4.2 使用 `RealDivList` 创建自定义列表

```
const divList = new RealDivList('myDivList', true, ['Item 1', 'Item 2', 'Item
3']);
divList.applyCSS('#myDivList', {
    '': { 'background-color': '#f0f0f0' },
    '>div': { 'padding': '10px', 'border-bottom': '1px solid #ccc' }
});
```

4.3 使用 `RealStory` 管理页面

```
RealStory.newPage().then(() => {
    console.log('Page 1 clear');
}).ofStory?.then?.(() => {
    console.log('Page 1 not clear');
});
```

5. 总结

`index.js` 提供了丰富的类和工具函数，能够帮助开发者更高效地处理浏览器环境中的 DOM 操作、事件处理、动画、文件加载等任务。通过合理使用这些类和函数，可以大大简化复杂的前端开发工作。