6 个提效开发的 JavaScript “杀手”函数 - 掘金

JavaScript 是 Web 开发中最关键的一环，提速 JS 开发就是提速下班[狗头]。

文章包含的代码片段，没有任何副作用，可以放心拷贝使用。

1. 校验一个元素是否在可视区域内

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。可以使用 `IntersectionObserver` 这个 API。

参考：IntersectionObserver API 使用教程- 阮一峰的网络日志

```
const callback = (entries) => {
  entries.forEach((entry) => {
    if (entry.isIntersecting) {
      // `entry.target` 是 dom 元素
      console.log(`${entry.target.id} is visible`);
    }
  });
};

const options = {
  threshold: 1.0,
};

const observer = new IntersectionObserver(callback, options);

const btn = document.getElementById( btn );
const bottomBtn = document.getElementById( bottom-btn );

observer.observe(btn);
observer.observe(bottomBtn);
```

`options` 参数能自定义 `Observer` 的行为。`threshold` 属性一般用的比较多，它定义的是 `Observer` 触发时，需要出现在可视区域中元素的可见百分比。

2. 识别设备

我们通常使用 `window.navigator.userAgent` 获取当前设备的细节来进行识别。

```
onst detectDeviceType = () =>
  /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera
Mini/i.test(
    navigator.userAgent
  )
    ?  Mobile
    :  Desktop ;

console.log(detectDeviceType());
```

3. 隐藏元素

CSS 隐藏元素通常有两种方法：

1. 可以使用 `style.visibility` 切换元素的可见性。
2. 如果想从整个渲染流中移除该元素，使用 `style.display` 属性。

```
const hideElement = (element, removeFromFlow = false) => {
  removeFromFlow
    ? (element.style.display =  none )
    : (element.style.visibility =  hidden );
};
```

如果不从渲染流中移除元素，只是隐藏可见性，元素仍然会被绘制，且占用视图空间。

当渲染长列表时，配合上方 `IntersectionObserver` 这个 API，使用 `style.display` 属性来隐藏不在可视区域内的元素，能较大提升渲染性能。

4. 获取 URL 上的 query 参数

推荐使用 `URL` 这个对象，`URL` 接口用于解析，构造，规范化和编码 URLs，用它可以很方便的获取链接上的 query 参数。

```
const url = new URL(window.location.href);
const paramValue = url.searchParams.get( paramName );
console.log(paramValue);
```

5. 简单的深拷贝

利用 `JSON` 方法先转化成 `string` 再转换为对象

```
const deepCopy = (obj) => JSON.parse(JSON.stringify(obj));
```

6. `wait` 方法

虽然我们有 `setTimeout` 方法来实现等待并异步执行，但是该方法不会返回 `Promise`，如果用在 `async` 函数中不是很方便，因此，我们可以自己实现一个 `wait` 方法。

```
const wait = (ms) => new Promise((resolve) => setTimeout(resolve,
ms));

const asyncFunc = async () => {
  await wait(1000);
  console.log( async );
};

asyncFunc();
```
