【前端面试】你用过scrollIntoView吗?

前言

不知道你有没有遇到这样的需求：锚点定位？进入页面某个元素需要出现在可视区？.....这一类的需求归根结底就是处理元素与可视区域的关系。我接触了很多前端小伙伴，实现的方式有各种各样的，比如使用 scrollTop、监听滚动等等，这也是很多小伙伴第一个想到的。

今天我们介绍一种更简单的方式：**scrollIntoView**。

1.基本介绍

scrollIntoView 是 HTMLElement 集合下的一个 API，每一个 HTML 元素都拥有这个 API。它的作用就和字面意思一样：**滚动到可视区。**为了更加官方一点，我们还是先来看看官方的解释。

官方解释：

> Element 接口的 scrollIntoView()方法会滚动元素的父容器，使被调用 scrollIntoView()的元素对用户可见。

官方的解释还是比较好理解的，我们摘要几个关键词出来：

- Element 接口
- 滚动
- 可见

从上我们大致也能理解，无非就是让某一个元素对用户可见，但是为了更准确又通俗一点，我们用自己的话简单概述一遍。

通俗解释：

> scrollIntoView()是 HTML 元素的一个方法，假如我们有一个元素容器出现了滚动条，有滚动条必然就有一些元素是不可见的，为了让隐藏的元素可见，我们可以手动滚动滚动条，让元素出现，另一种方法就是调用隐藏元素的 scrollIntoView 方法，让它自动滚动到可视区内。

2.基本使用

基本的概念我们了解清楚了，那么接下来我们在看看这个方法如何使用?

语法：

```
element.scrollIntoView(); // 等同于 element.scrollIntoView(true)
element.scrollIntoView(alignToTop); // Boolean 型参数
element.scrollIntoView(scrollIntoViewOptions); // Object 型参数
```

参数解释：

- alignToTop：它是一个 Boolean 值，它用来规定元素出现在可视区后与可视区的对齐方式，为 true 代表顶端对齐，false 代表低端对齐。
- scrollIntoViewOptions：它是一个对象，该参数主要是配置元素的动画效果以及位置的，它有一下 3 个属性：
  - behavior：它定义元素出现在可视区内过程的动画，有 auto 和 smooth 两种选择。
  - block：定义元素的垂直方向的对齐方式，有”start”, ”center”, ”end”, 或 ”nearest” 4 个选项，默认 start。
  - inline：定义元素水平对齐方式，有”start”, ”center”, ”end”, 或 ”nearest”4 个选项，默认”nearest”。

注意：

有些小伙伴可能发现两个参数都能定义元素的对齐方式，它们之间有什么联系呢？当 alignToTop 为 true 时，scrollIntoViewOptions: {block: ”start”, inline: ”nearest”}这是它的默认值，当 alignToTop 为 false 时，scrollIntoViewOptions: {block: ”end”, inline: ”nearest”}这是它的默认值。

3.代码实战

知道了 scrollIntoView 的作用，那么在实际场景中如何使用呢？接下我们就以最常见的锚点需求来做演示。

实现目标：

点击左侧标题，右侧对应内容平滑出现在可视区内。

HTML 代码：

html 代码很简单，就是一个很普通的左右布局结构，左侧列表，右侧具体内容，**代码如下：**

```
<body>
  <div class="box">
    <div class="left">
      <p>第一段内容</p>
      <p>第二段内容</p>
      <p>第三段内容</p>
    </div>
    <div class="right">
      <div class="content">
        我是第一段内容
      </div>
      <div class="content">
```

```
        我是第二段内容
      </div>
      <div class="content">
        我是第三段内容
      </div>
    </div>
  </div>
</body>
```

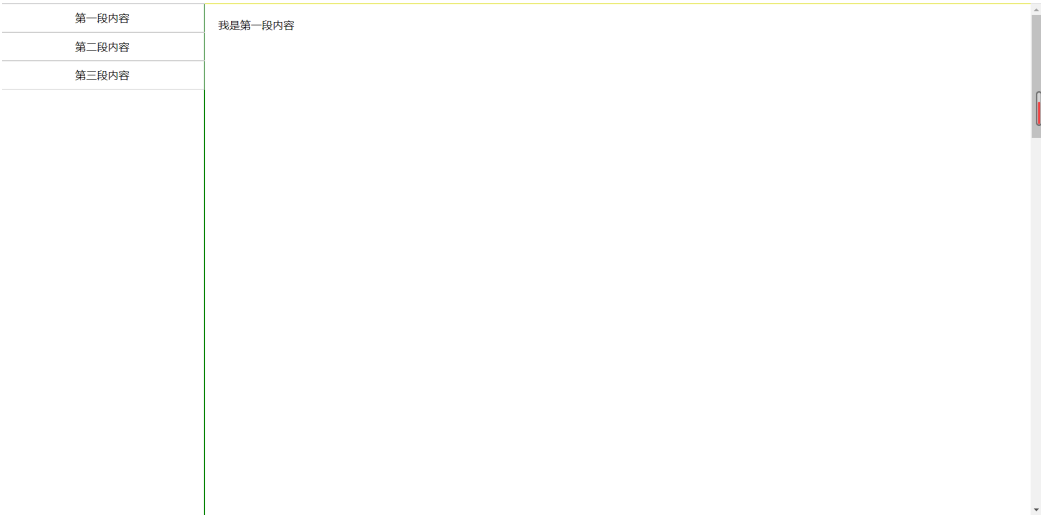
CSS 代码：

添加布局样式，**代码如下**：

```
<style>
  * {
    margin: 0;
    padding: 0;
  }

  .box {
    width: 100vw;
    height: 100vh;
    display: flex;
    overflow: hidden;
  }

  .left {
    width: 300px;
    height: 100%;
    border-right: 2px solid green;
  }

  p {
    height: 40px;
    width: 100%;
    display: flex;
    align-items: center;
    justify-content: center;
    cursor: pointer;
    border: 1px solid #ccc;
  }

  .right {
    flex: 1;
    overflow: auto;
  }

  .content {
    width: 100%;
    height: 1000px;
    border-top: 1px solid yellow;
    padding: 20px;
```

```
    box-sizing: border-box;
  }
</style>
```

布局效果：

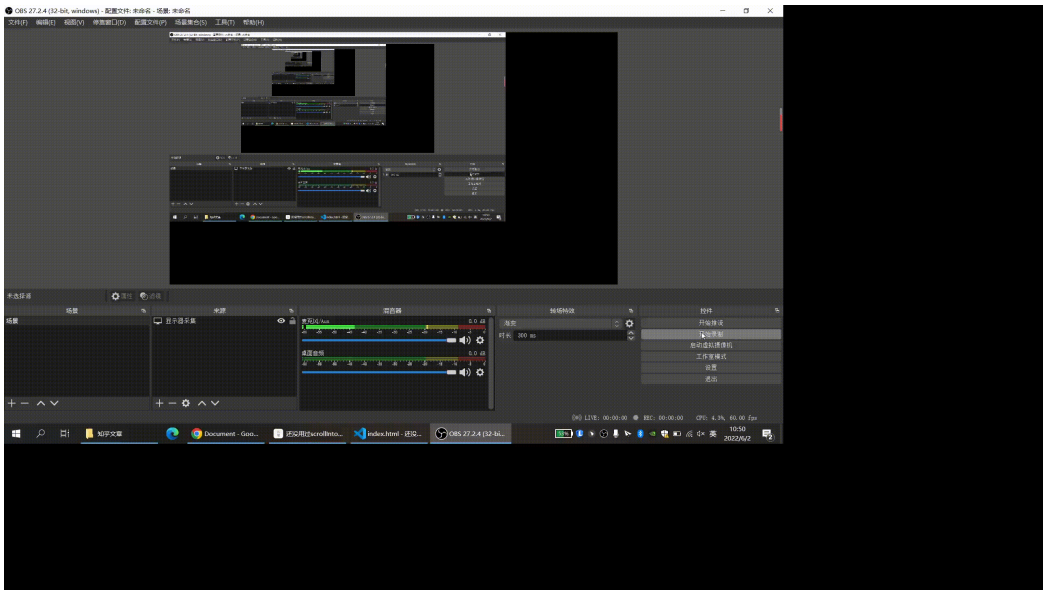


JS 代码：

我们的代码实现思路非常的简单：首先给每个标题添加点击事件，当点击标题时，对应的内容元素调用 scrollIntoView 方法出现在可视区内，**代码如下：**

```
<script>
  // 获取三个标题元素
  let pDomList = document.getElementsByTagName('p');
  // 获取三段内容元素
  let contentDomList = document.getElementsByClassName('content');

  // 添加点击事件
  for (let index = 0; index < pDomList.length; index++) {
    pDomList[index].addEventListener('click', () => {
      contentDomList[index].scrollIntoView({
        behavior: 'smooth',
        block: 'start',
        inline: 'start'
      })
    })
  }
</script>
```

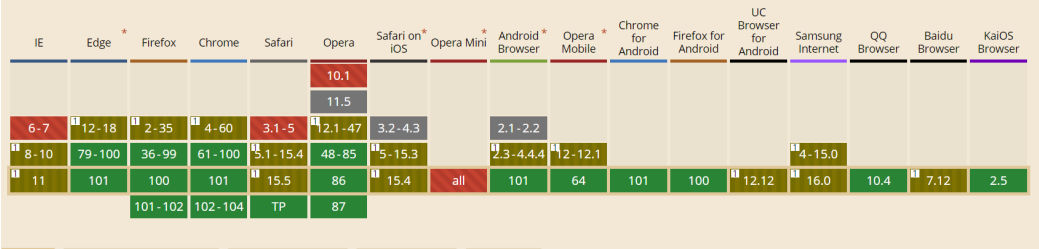
最终结果：



上图中的效果我相信有很多场景都会用到，当然实现的办法有很多，比如使用 a 标签锚点定位等等，但是当不能用 a 标签的时候，我们建议使用 scrollIntoView。

4.兼容性

使用一个 API 的时候，我们一定要注意它的兼容性，一起来看看：



可以看到这个 API 的兼容性还是非常好的，连 IE 都能兼容。

总结

HTMLElement 集合下的 API 非常多，很多时候我们当我们遇到一个问题不能解决的时候，不妨回过头看看基础知识，说不定有你意想不到的惊喜。