

自定义的window.onload事件不触发

原因:

- 1、window.onload事件在异步回调里定义
- 2、在定义window.onload事件之前，浏览器的资源已经加载完了。

示例1:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10     
11
12     <script type="text/javascript">
13
14         function A(){
15             setTimeout(function(){
16                 console.log("loading");
17                 window.addEventListener("load", function(){
18                     console.log("load1");
19                 });
20             }, 0);
21
22             window.addEventListener("load", function(){
23                 console.log("load2");
24             });
25         };
26
27         A();
28     </script>
29 </body>
30 </html>
```

结果: 打印load1

示例2:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
```

```
10     <script type="text/javascript">
11
12         function A(){
13             setTimeout(function(){
14                 console.log("loading");
15                 window.addEventListener("load", function(){
16                     console.log("load1");
17                 });
18             }, 0);
19
20             window.addEventListener("load", function(){
21                 console.log("load2");
22             });
23         };
24
25         A();
26     </script>
27 </body>
28 </html>
```

结果：不打印load1

结论：

两者的区别在于前者加载了一个img，加载img是异步操作，setTimeout回调执行的时候img还未加载完，因此定义的onload可以在后续触发。

后者没有加载img，当setTimeout的回调执行的时候，onload已经触发过了，此时定义的onload事件没有机会触发了。

posted @ 2022-12-16 10:37 胡姐姐 阅读(430) 评论(0)