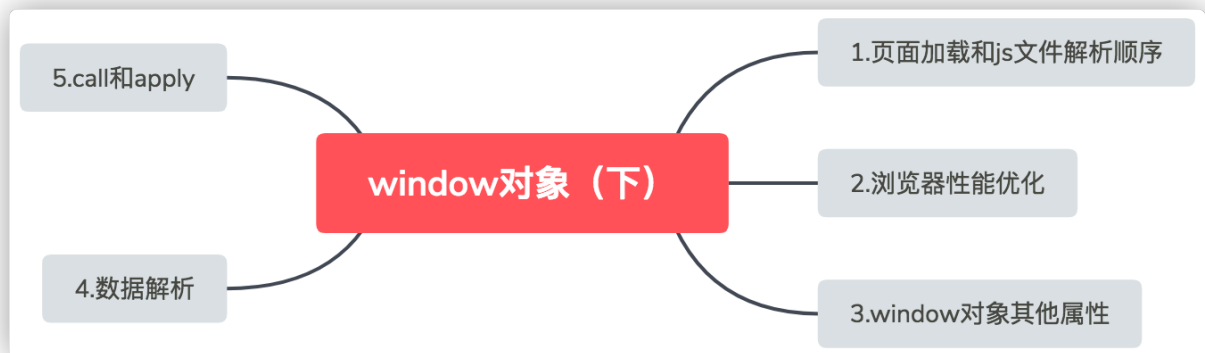


window对象(下)

0.1. 本节主要内容：



0.1. 学习目标：

节数	知识点	要求
第一节 页面加载和js文件解析顺序	一个页面的加载过程	了解
	defer和async	了解
第二节 浏览器性能优化	回流和重绘	了解
	documentFragment概述	了解
第三节 window对象其他属性	history对象	掌握
第四节 数据解析	数据解析	掌握
第五节 call和apply	call的使用	掌握
	apply的使用	掌握
	call和apply的区别	掌握

1. 页面加载和js文件解析顺序

1.1. 服务器端和客户端

1.2. 一个页面的加载过程

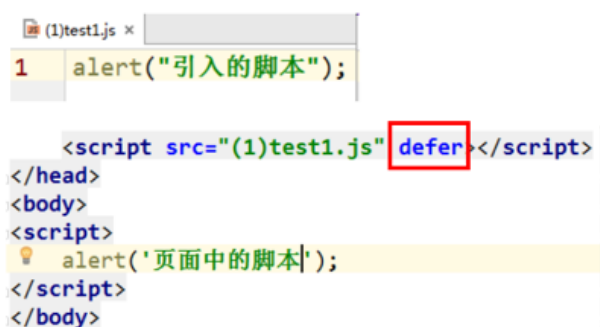
- 1.用户输入网址（假设是个html页面，并且是第一次访问），浏览器向服务器发出请求，服务器返回html文件；
 - 2.浏览器开始载入html代码，发现<head>标签内有一个<link>标签引用外部CSS文件；
 - 3.浏览器又发出CSS文件的请求，服务器返回这个CSS文件；
 - 4.浏览器继续载入html中<body>部分的代码，并且CSS文件已经拿到手了，可以开始渲染页面了；
 - 5.浏览器在代码中发现一个标签引用了一张图片，向服务器发出请求。此时浏览器不会等到图片下载完，而是继续渲染后面的代码；
 - 6.服务器返回图片文件，由于图片占用了一定面积，影响了后面段落的排布，因此浏览器需要回过头来重新渲染这部分代码；
 - 7.浏览器发现了一个包含一行Javascript代码的<script>标签，赶快运行它；
- Javascript脚本执行了这条语句，它命令浏览器隐藏掉代码中的某个<div>（style.display="none"）。茶具啊，突然就少了这么一个元素，浏览器不得不重新渲染这部分代码；
- 8.终于等到了</html>的到来，浏览器泪流满面.....
 - 9.等等，还没完，用户点了一下界面中的“换肤”按钮，Javascript让浏览器换了一下<link>标签的CSS路径；

10.浏览器召集了在座的各位<div> 们，“大伙儿收拾收拾行李，咱得重新来过.....”，浏览器向服务器请求了新的CSS文件，重新渲染页面。

1.3. defer

defer属性：等待DOM加载完成后才去加载JS脚本

例：



```
(1)test1.js x
1 alert("引入的脚本");

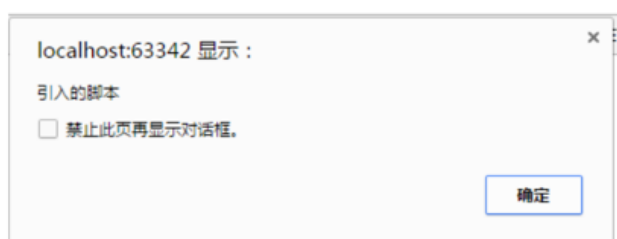
<script src="(1)test1.js" defer</script>
</head>
<body>
<script>
  alert('页面中的脚本');
</script>
</body>
```

A red arrow points from the `defer` attribute in the code to the first dialog box.

第一次弹窗：



第二次弹窗：



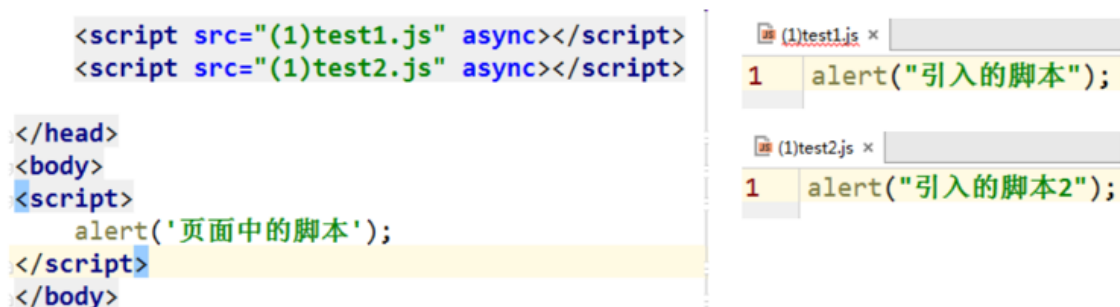
1.4. async

async属性：DOM加载和js脚本加载异步执行，同时进行。

async优势：避免了因DOM文件过大导致的【文件加载阻塞】

async缺陷：无法确定js脚本到底何时执行，并仅对外部js脚本生效

例：



按照我们关于async的说法，此时我们引入的两个js外部脚本应该是和本地dom异步执行的。也就是说究竟弹窗先弹出哪一个alert都是不确定的。但是事实却似乎并不是这样：



结果：

脚本的实际执行却总是按照脚本加载时出现的顺序执行的，每一次的结果都是如左面展示的顺序。

原因：

HTML5规范要求
脚本执行应该按照脚本出现的先后顺序执行

1.5. 不添加任何属性的js脚本

js脚本在不受任何外界因素影响的时候，实际上就是按照html代码的加载顺序执行的。因此【不添加任何属性的js脚本】总是会在【引入脚本的节点】加载完毕之前执行。



2. 浏览器性能优化

2.1. 什么是回流

当render tree中的一部分(或全部)因为元素的规模尺寸，布局，隐藏等改变而需要重新构建。**这就称为回流**。每个页面至少需要一次回流，就是在页面第一次加载的时候。

2.2. 什么是重绘

当render tree中的一些元素需要更新属性，而这些属性只是影响元素的外观，风格，而不会影响布局的，比如background-color。则就叫称为重绘。

注：回流必将引起重绘，而重绘不一定会引起回流。

2.3. 引起重绘和回流的原因

1.页面初始渲染

2.改变字体，改变元素尺寸（宽、高、内外边距、边框，改变元素位置等）

（注意：如果修改属性不影响布局则不会发生重排）

3..改变元素内容（文本或图片等或比如用户在input框中输入文字）

4.添加/删除可见DOM元素（注意：如果是删除本身就display:none的元素不会发生重排；visibility:hidden的元素显示或隐藏不影响重排）

5.fixed定位的元素,在拖动滚动条的时候会一直回流

6.调整窗口大小 (Resizing the window)

7.计算 offsetWidth 和 offsetHeight 属性

2.4. 如何从重绘和回流方面提高浏览器性能

1.不要一项一项的去改变样式，尽可能一口气写完。(可以写在一起，不要被打断就行)最好使用.style.cssText

2.读写DOM也尽量也放在一起

3.使用文档碎片 var linshi =
document.createDocumentFragment();

4.使用fixed或者absolute可以减少回流和重绘

2.5. DocumentFragment简单描述

documentFragment是nodeType值为11，nodeName的值为#document-fragment.

1) documentFragment是一个文档片段，一种‘轻量级节点’

2) 通常作为仓库来使用，不存在DOM树上，是一种游离态

DocumentFragment的用途

当我们用JS的DOM创建很多节点时，在加入节点到DOM树上时，节点需要一个个渲染，这样节点数较多时就会影响浏览器的渲染效率，这个时候我们将创建的节点都放在**DocumentFragment**这样的节点上，然后把DocumentFragment加入至DOM，只需要完成一次渲染就可以达到之前很多次渲染的效果！！！！

举个栗子（创建100个li元素内容为1-100）：

```
var ul = document.createElement('ul');

var flag = document.createDocumentFragment();

for(var i=1; i<101;i++){

var li = document.createElement('li')

var liText = document.createTextNode(i);

li.appendChild(liText);

flag.appendChild(li);

}

ul.appendChild(flag);

document.body.appendChild(ul);
```

2.6. 实例：带有提示的滑动条

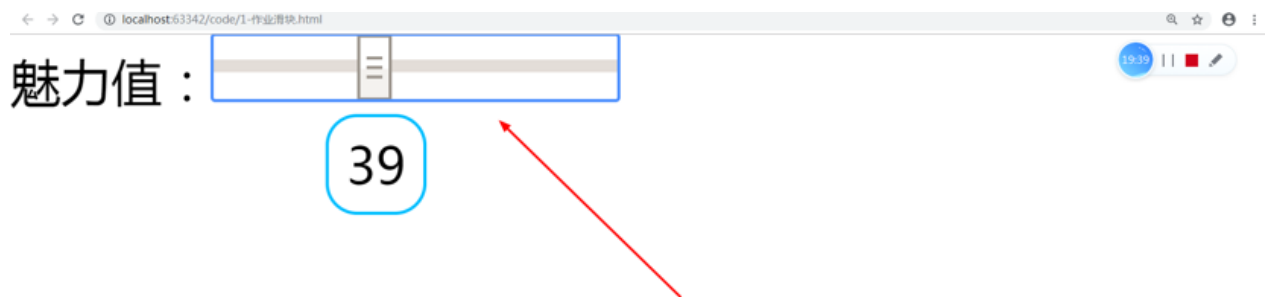
分析一下：

主要有三个事件：onmousedown,onmousemove,onmouseup


```
1-作业滑块.html × 18:42
html body script
4 <meta charset= UTF-8 >
5 <title></title>
6 <style>
7 *{
8     margin:0;
9     padding:0;
10 }
11 .box{
12     width: 30px;
13     height: 30px;
14     border:1px solid deepskyblue;
15     border-radius: 10px;
16     line-height: 30px;
17     text-align: center;
18     display: none;
19     position: relative;
20 }
21
```

```
24 <body>
25 <span>魅力值: </span><input type="range" value="0" max="100" min="0" step="1" class
26 <div class="box"></div>
```

```
1-作业滑块.html × 19:20
html body script
28 //分别获取范围值元素和魅力值元素
29 var orange = document.querySelector(".range");
30 document.querySelector(".box");
31 var flag = false; //开关默认是关闭的
32 orange.onmousedown = function() {
33     obox.style.display = "block";
34     obox.innerHTML = orange.value;
35 }
36
37
38 orange.onmousemove = function() {
39     obox.innerHTML = orange.value;
40     obox.style.left = event.clientX -15 + "px";
41 }
42
43 orange.onmouseup = function() {
44
45     obox.style.display = "none";
46 }
```



2.7. 实例：改变字体大小和颜色

1、获取select中option中的值



2、改变字体大小和颜色

```
13-sh字体颜色改变.html x
html body script
10 <body>
11 字体颜色:
12 <select name="" id="color">
13   <option value="0">=请选择颜色=</option>
14   <option value="1">红色</option>
15   <option value="2">蓝色</option>
16   <option value="3">绿色</option>
17 </select>
18 字体大小:
19 <select name="" id="font">
20   <option value="0">=请选择字体大小=</option>
21   <option value="1">20</option>
22   <option value="2">30</option>
23   <option value="3">80</option>
24 </select>
```

```
13-sh字体颜色改变.html x
html body script
25 <div class="box">
26   一开始是朱德提出来的,
27   本来只有前面十六个字, 简称十六字方针。
28   那是中国工农红军在土地革命战争时期进行
29   游击战争的作战指导原则, 即“敌进我退, 敌驻我扰, 敌疲
30   我打, 敌退我追”, 简称“十六字诀”。“十六字诀”包举了反“围
31   剿”作战的基本原则, 包举了战略防御和战略进攻两个阶段, 在战略防御时又
32   包举了战略退却和战略反攻的两个阶段。“十六字诀”的基本
33   精神是, 从敌大我小、敌强我弱的基本特点出发, 利用根据地创造的有
34   利形势, 灵活地使用兵力和变换战法, 趋利避害, 扬长击短, 在战争中务求保存和发
35   展自己, 以多打胜仗、消灭敌人的办法, 逐渐改变敌我形势, 夺取战争的胜利。
36
37 </div>
```

```
13-sh字体颜色改变.html x
html body script
38 <script>
39   var ocolor = document.querySelector("#color");//获取select元素
40   var ofont = document.querySelector("#font");
41   var obox = document.querySelector(".box");
42   ocolor.onchange = changeColor;
43   // ofont.onchange = changeFontSize;
44   ofont.addEventListener("change", changeFontSize);
```

```
13-sh字体颜色改变.html x
html body script
45 function changeColor() {
46     switch(ocolor.value) {
47         case "1":
48             obox.style.color = "red";
49             break;
50         case "2":
51             obox.style.color = "blue";
52             break;
53         case "3":
54             obox.style.color = "green";
55             break;
56     }
57 }
58
```

```
13-sh字体颜色改变.html x
html body script
59 function changeFontSize() {
60     switch(ofont.value) {
61         case "1":
62             obox.style.fontSize = "20px";
63             break;
64         case "2":
65             obox.style.fontSize = "30px";
66             break;
67         case "3":
68             obox.style.fontSize = "80px";
69             break;
70     }
71 }
72
73 </script>
```

localhost:63342/js-事件-day9/code/13-sh字体颜色改变.html

字体颜色: 红色 字体大小: 30

一开始是朱德提出来的，本来只有前面十六个字，简称十六字方针。那是中国工农红军在土地革命战争时期进行游击战争的作战指导原则，即“敌进我退，敌驻我扰，敌疲我打，敌退我追”，简称“十六字诀”。“十六字诀”包举了反“围剿”作战的基本原则，包举了战略防御和战略进攻两个阶段，在战略防御时又包举了战略退却和战略反攻的两个阶段。“十六字诀”的基本精神是，从敌大我小、敌强我弱的基本特点出发，利用根据地创造的有利形势，灵活地使用兵力和变换战法，趋利避害，扬长击短，在战争中务求保存和发展自己，以多打胜仗、消灭敌人的办法，逐步改变敌强我弱的形势，夺取战争的胜利。

3. window对象其他属性

3.1. history对象

window.history对象表示整个浏览器的页面栈对象。在对象中提供了一些属性和方法来帮助更好的控制整个浏览器中页面的访问。

(1)window.history.back() 跳转到栈中的上一个页面

(2)window.history.forward() 跳转到栈中的下一个页面

(3)window.history.go(num) 跳转到栈中的指定页面

3.2. 存储对象

JavaScript 存储对象

Web 存储 API 提供了 sessionStorage（会话存储）和 localStorage（本地存储）两个存储对象来对网页的数据进行添加、删除、修改、查询操作。

sessionStorage：用于临时保存同一窗口(或标签页)的数据，在关闭窗口或标签页之后将会删除这些数据。

localStorage：用于长久保存整个网站的数据，保存的数据没有过期时间，直到手动去除。

保存数据：setItem(key,value)

读取数据：getItem(key)

删除单个数据：removeItem(key)

删除所有数据：clear()

得到某个索引的key(index)

```
localStorage.setItem('username','xiaoming');
localStorage.password = '123456';

document.body.innerHTML = "username:" +
localStorage.getItem('username');
document.body.innerHTML += "<br>password:" +
localStorage.password
```

```
sessionStorage.username = "xiaoming123";
sessionStorage.password = "123456123";
document.body.innerHTML = "username:" +
sessionStorage.getItem('username');
document.body.innerHTML += "<br>password:" +
sessionStorage.password
```

4. 数据解析

数据解析：将【不能被直接使用的数据】通过某种方法转变为【能够被直接使用的数据】的过程称为数据解析。

而对于开发者来说最常见的数据解析就是将【字符串数据】解析为【对象数据】

例如：

假设我们得到了一个字符串数据为：

```
var data = "?name=frank&age=18&sex=male";
```

很显然这样一个字符串是没有办法直接为我们所用，因此我们可以通过如下的手段将这个字符串变更为一个对象。

```
var infoArr = window.location.search.slice(1).split("&"); var obj = {};  
for(var i=0;i<infoArr.length;i++){ var temp = infoArr[i].split("=");  
if(temp[0]){obj[temp[0]] = temp[1];} }
```

表单数据解析

将表单中的数据先转换成数组，再转换成对象，最后将数据添加到对象中

5. call和apply

调用方法，替换对象。

- 1、每个函数都包含两个非继承而来的方法：call()方法和apply()方法。
- 2、call()、apply()都是用来重定义 **this** 这个对象的
- 3、相同点：这两个方法的作用是一样的。

都是在特定的作用域中调用函数，等于设置函数体内**this**对象的值，以扩充函数赖以运行的作用域。

一般来说，this总是指向调用某个方法的对象，但是使用call()和apply()方法时，就会改变this的指向。

5.1. call的使用

这两个方法就是用来调用函数的。语法如下：

call(对象,[形参, 形参])

```
<script>
    window.color = 'red';
    document.color = 'yellow';

    var s1 = {color: 'blue' };
    function changeColor(){
        console.log(this.color);
    }

    changeColor.call();           //red (默认传递参数)
    changeColor.call(window);    //red
    changeColor.call(document);  //yellow
    changeColor.call(this);      //red
    changeColor.call(s1);        //blue

</script>
```

```
//例2
var Pet = {
    words : '...',
    speak : function (say) {
        console.log(say + ' ' + this.words)
    }
}
Pet.speak('Speak'); // 结果: Speak...

var Dog = {
    words: 'Wang'
}

//将this的指向改变成了Dog
Pet.speak.call(Dog, 'Speak'); //结果: SpeakWang
```

5.2. apply的使用

apply(对象,[array(形参数组)])


```

<script>
    window.number = 'one';
    document.number = 'two';

    var s1 = {number: 'three' };
    function changeColor(){
        console.log(this.number);
    }

    changeColor.apply();           //one (默认传参)
    changeColor.apply(window);    //one
    changeColor.apply(document);  //two
    changeColor.apply(this);      //one
    changeColor.apply(s1);        //three

</script>

```

```

//例2
function Pet(words){
    this.words = words;
    this.speak = function () {
        console.log( this.words)
    }
}

function Dog(words){
    //Pet.call(this, words); //结果: Wang
    Pet.apply(this, arguments); //结果: Wang
}

var dog = new Dog('Wang');
dog.speak();

```

5.3. call和apply的区别

从定义中可以看出，call和apply都是调用一个对象的一个方法，用另一个对象替换当前对象。而不同之处在于传递的参数，apply最多只能有两个参数——新this对象和一个数组argArray，如果arg不是数组则会报错TypeError；

call则可以传递多个参数，第一个参数和apply一样，是用来替换的对象，后边是参数列表。

实例：

1、防抖:高频事件触发时防止高频次执行特定程序

当高频事件触发时，只有在两次高频事件间隔一定时间后才会执行特定程序。

如果n秒内高频事件再次发生，则重新计时。

思路：每次事件触发时，都取消上次等待执行的特定程序，重新等待。需要使用延时函数。

```
<body>
  <input type="text" name="" id="" value="" />
</body>
```

```
<script type="text/javascript">
  var ipt = document.querySelector("input");
  ipt.oninput = pr(work,500);

  function pr(fn,delay){
    var timer = null;
    return function(){
      clearTimeout(timer);
      timer = setTimeout(function(){
        // fn();
        // fn.apply(this,[11,22,33])
        fn.call(this,11,22,33);
      },delay)
    }
  }

  function work(a,b,c){
    console.log(a,b,c);
    console.log(ipt.value);
  }
</script>
```

2、节流

当高频事件触发时，在n秒内只会执行特定程序一次，本次不执行完成，不会执行下一次。

节流会稀释特定程序被执行的次数。

思路：每次触发执行特定程序前，都先判断一下是否有等待执行的特定程序，如果有，则本次不执行特定程序。需要使用延时函数。

```
function pr(fn){
    var isRunnging = false;
    return function(){
        if (isRunnging) {
            return;
        }
        isRunnging = true;
        setTimeout(=>{
            // fn();
            fn.apply(this);
            isRunnging = false;
        },3000);
    }
}

window.onresize = pr(work);

function work(){
    console.log(window.outerWidth)
    console.log(window.outerHeight)
}
```