设计模式

event loop事件循环

let a = new Set([1, 2, 3]);

let b = new Set([3, 5, 2]);

// 并集

let unionSet = new Set([...a, ...b]);

//[1,2,3,5]

// 交集

let intersectionSet = new Set([...a].filter(x => b.has(x)));

// [2,3]

// ab差集

let differenceABSet = new Set([...a].filter(x => !b.has(x)));

react父子组件的渲染顺序

父componentWillMount

父render

子componentWillMount

子render

子componentDidMount

父componentDidMount

父render

子componentWillReceiveProps

子render

Vue和React的数据绑定对比以单向数据流和单双向数据绑定

https://www.jianshu.com/p/6e124ad23c68

https://www.zhihu.com/question/31585377

在数据绑定上来说，vue的特色是双向数据绑定，而在react中是单向数据绑定。表单的双向绑定，说到底不过是 （value 的单向绑定 + onChange 事件侦听）的一个语法糖

Vue和React遵循从上到下的数据流向，即单向数据流。

Vue 双向数据绑定实现

function defineReactive(obj, key, value){

var dep = new Dep();

Object.defineProperty(obj, key, {

enumerable: true,

configurable: true,

get: function reactGetter(){

if(Dep.target){

dep.depend();

}

return value;

},

set: function reactSetter(newVal){

if (value === newVal) {

return;

} else {

value = newVal;

//如果数据发生改变，则通知所有的 watcher（借助 dep.notify()）

dep.notify();

}

}

})

}

柯里化函数

function add(n) {

var fn = function(m) {

return add(n + m);

};

fn.valueOf = function() {

return n;

};

fn.toString = function() {

return '' + n;

};

return fn;

}

add(1) // 结果 1

add(1)(2) // 结果 3

add(1)(2)(3) // 结果 6

add(1)(2)(3)(4) // 结果 10

js文件加载阻塞以及defer和async的区别

<script src="script.js"></script>

没有 defer 或 async，浏览器会立即加载并执行指定的脚本，“立即”指的是在渲染该 script 标签之下的文档元素之前，也就是说不等待后续载入的文档元素，读到就加载并执行

<script async src="script.js"></script>

有 async，加载和渲染后续文档元素的过程将和 script.js 的加载与执行并行进行（异步）。

<script defer src="myscript.js"></script>

有 defer，加载后续文档元素的过程将和 script.js 的加载并行进行（异步），但是 script.js 的执行要在所有元素解析完成之后，DOMContentLoaded 事件触发之前完成。

css加载是否会阻塞页面渲染

1、css加载不会阻塞DOM树的解析

2、css加载会阻塞DOM树的渲染

3、css加载会阻塞后面js语句的执行

为了避免让用户看到长时间的白屏时间，我们应该尽可能的提高css加载速度，比如可以使用以下几种方法:

使用CDN(因为CDN会根据你的网络状况，替你挑选最近的一个具有缓存内容的节点为你提供资源，因此可以减少加载时间)

对css进行压缩(可以用很多打包工具，比如webpack,gulp等，也可以通过开启gzip压缩)

合理的使用缓存(设置cache-control,expires,以及E-tag都是不错的，不过要注意一个问题，就是文件更新后，你要避免缓存而带来的影响。其中一个解决防范是在文件名字后面加一个版本号)

减少http请求数，将多个css文件合并，或者是干脆直接写成内联样式(内联样式的一个缺点就是不能缓存)

‘abc1234def5678gh’变成abc,1,2,3,4,def,5,6,7,8,gh

var s="abc1234def5678gh";

s.replace(/\d{1}/g,function(s1){return ' ' + s1 + ' '}).trim().replace(/\s+/g,'，');

//"abc,1,2,3,4,def,5,6,7,8,gh"

var s="abcA1234def5678gh";

s.replace(/[a-zA-Z]{1}/g,function(s1){return ' ' + s1 + ' '}).trim().replace(/\s+/g,',');

//"a,b,c,A,1234,d,e,f,5678,g,h"

堆栈

https://juejin.im/post/5ae09017f265da0b8f623aa3

正则表达式后行断言 • 探索 ES2018 和 ES2019

var s="12345678";

s.replace(/(?=(\d{3})+(?!\d))/g,",");

//"12,345,678"

https://juejin.im/entry/5b4808e9f265da0f9c678ff4

权限控制

https://juejin.im/post/5c0a96e66fb9a049ec6ae305

安全

https://juejin.im/entry/584b880a8e450a006c590576

缓存

https://juejin.im/post/5b0ea4f1518825155d66a97b

https://juejin.im/post/5ae081aaf265da0b767d263a

如何实现浏览器内多个标签页之间的通信?(阿里)

WebSocket、SharedWorker；

也可以调用localstorge、cookies等本地存储方式；

localstorge另一个浏览上下文里被添加、修改或删除时，它都会触发一个事件，

我们通过监听事件，控制它的值来进行页面信息通信；

注意quirks：Safari 在无痕模式下设置localstorge值时会抛出QuotaExceededError 的异常；

请描述一下cookies，sessionStorage和localStorage 的区别？

cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。

cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递。

sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

存储大小：

cookie数据大小不能超过4k。

sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。

有期时间：

localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；

sessionStorage 数据在当前浏览器窗口关闭后自动删除。

cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭

html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分HTML 和HTML5？

\* HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。

绘画 canvas;

用于媒介回放的 video 和 audio 元素;

本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失;

sessionStorage 的数据在浏览器关闭后自动删除;

语意化更好的内容元素，比如 article、footer、header、nav、section;

表单控件，calendar、date、time、email、url、search;

新的技术webworker, websocket, Geolocation;

移除的元素：

纯表现的元素：basefont，big，center，font, s，strike，tt，u;

对可用性产生负面影响的元素：frame，frameset，noframes；

\* 支持HTML5新标签：

IE8/IE7/IE6支持通过document.createElement方法产生的标签，

可以利用这一特性让这些浏览器支持HTML5新标签，

浏览器支持新标签后，还需要添加标签默认的样式。

当然也可以直接使用成熟的框架、比如html5shim;

<!--[iflt IE 9]>

<script>src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>

<![endif]-->

\* 如何区分HTML5： DOCTYPE声明\新增的结构元素\功能元素

介绍一下你对浏览器内核的理解？

主要分成两部分：渲染引擎(layoutengineer或RenderingEngine)和JS引擎。

渲染引擎：负责取得网页的内容（HTML、XML、图像等等）、整理讯息（例如加入CSS等），以及计算网页的显示方式，然后会输出至显示器或打印机。

浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不相同。所有网页浏览器、电子邮件客户端以及其它需要编辑、显示网络内容的应用程序都需要内核。

JS引擎则：解析和执行javascript来实现网页的动态效果。

最开始渲染引擎和JS引擎并没有区分的很明确，后来JS引擎越来越独立，内核就倾向于只指渲染引擎。

移动端兼容问题

1、input 有placeholder情况下不要设置行高，否则文本会偏上

2、input为fixed定位在ios下的bug问题，点击焦点input，input跳到中间位置

软键盘唤起后，页面的 fixed 元素将失效（即无法浮动，也可以理解为变成了 absolute 定位），所以当页面超过一屏且滚动时，失效的 fixed 元素就会跟随滚动了

既然在 iOS 下由于软键盘唤出后，页面 fixed 元素会失效，导致跟随页面一起滚动，那么假如——页面不会过长出现滚动，那么即便 fixed 元素失效，也无法跟随页面滚动，也就不会出现上面的问题了。

那么按照这个思路，如果使 fixed 元素的父级不出现滚动，而将原 body 滚动的区域域移到 main 内部

3、一些情况下对非可点击元素如(label,span)监听click事件，ios下不会触发，css增加cursor:pointer就搞定了

4、new Date('2019-01-30')在IOS上无效，使用moment转一次

5、关于 iOS 系统中，中文输入法输入英文时，字母之间可能会出现一个六分之一空格；

this.value = this.value.replace(/\u2006/g,'');

6、手机上的flex布局时会有兼容性问题，只用新版本的会出现安卓手机不识别的现象

.box{

display: -webkit-box;

/\* 老版本语法: Safari, iOS, Android browser, older WebKit browsers. \*/

display: -moz-box; /\* 老版本语法: Firefox (buggy) \*/

display: -ms-flexbox; /\* 混合版本语法: IE 10 \*/

display: -webkit-flex; /\* 新版本语法: Chrome 21+ \*/

display: flex; /\* 新版本语法: Opera 12.1, Firefox 22+ \*/

}

事件流

JavaScript中的事件流分为捕获阶段、目标阶段、冒泡阶段，我们可以JS控制需要操作捕获还是冒泡阶段的事件。

根据事件流的原理，我们可以优化事件处理方法，比如进行事件委托。

“DOM2级事件”定义了两个方法，用于处理指定和删除事件处理程序的操作：addEventListener( )和removeEventListener( )。

所有的DOM节点中都包含这两个方法，并且它们接收3个参数：要处理的事件，作为事件处理程序的函数和一个布尔值。布尔值如果是true，表示在捕获阶段调用事件处理程序，如果是false，表示在冒泡阶段调用事件处理程序。

输入框连续搜索请求两次，第二次结果先到，怎么取消第一次的搜索结果渲染？

方法一：后台将搜索关键词一并返给前端，前端通过判断是否为当前搜索关键词决定是否渲染。

方法二：

var CancelToken = axios.CancelToken;

var source = CancelToken.source();

axios.get('/user/12345', {

cancelToken: source.token

}).then(()=>{

// 你的逻辑

}).catch(function(thrown) {

// 如果调用了cancel方法，那么这里的res就是cancel传入的信息

// 你的逻辑

});

source.cancel('这里你可以输出一些信息，可以在catch中拿到');

方法三：

var native = new XMLHttpRequest();

native.open("GET","https://api.github.com/");

native.send();

native.onreadystatechange=function(){

if(native.readyState==4&&native.status==200){

console.log(native.response);

}else{

console.log(native.status);

}

}

native.abort();

算法排序

1、冒泡排序

function bubbleSort(arr) {

　　var len = arr.length;

　　for (var i = 0; i < len; i++) {

　　　　for (var j = 0; j < len - 1 - i; j++) {

　　　　　　if (arr[j] > arr[j+1]) { //相邻元素两两对比

　　　　　　　　[arr[j], arr[j+1]] = [arr[j+1], arr[j]]

　　　　　　}

　　　　}

　　}

　　return arr;

}

var arr=[3,44,38,5,47,15,36,26,27,2,46,4,19,50,48];

console.log(bubbleSort(arr));

2、插入排序

function insertionSort(array) {

　　for (var i = 1; i < array.length; i++) {

　　　　var key = array[i];

　　　　var j = i - 1;

　　　　while ( array[j] > key) {

　　　　　　array[j + 1] = array[j];

　　　　　 j--;

　　　　}

　　　　array[j + 1] = key;

　　}

　　return array;

}

var arr=[3,44,38,5,47,15,36,26,27,2,46,4,19,50,48];

console.log(insertionSort(arr));

3、快速排序

function quickSort(arr) {

　　if (arr.length <= 1) { return arr; }

　　var pivotIndex = Math.floor(arr.length / 2);

　　var pivot = arr.splice(pivotIndex, 1)[0];

　　var left = [];

　　var right = [];

　　for (var i = 0; i < arr.length; i++){

　　　　if (arr[i] < pivot) {

　　　　　　left.push(arr[i]);

　　　　} else {

　　　　　　right.push(arr[i]);

　　　　}

　　}

　　return quickSort(left).concat([pivot], quickSort(right));

};

var arr=[3,44,38,5,47,15,36,26,27,2,46,4,19,50,48];

console.log(quickSort2(arr));

react父组件怎么触发子组件方法

方法一：

<Child onRef={this.onRef} />

this.props.onRef(this)

子组件将this传递给父组件，父组件获取到后触发子组件this上的方法

方法二:

通过componentWillReceiveProps来判断属性是否改变然后触发子组件方法