



真传X

IT前沿技术在线大学

www.zhenchuanx.com

基础知识 (二)

从设计模式说起

<https://coolshell.cn/articles/4535.html>

软件工程设计原则

1. 高内聚
2. 低耦合

设计模式的历史和相关资料

- “四人帮” (Gang of Four, 简称 GoF)
- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
- Design Patterns - Elements of Reusable Object-Oriented Software
- 1991 年
- 23 种 ?
- [wiki](#)
- [JavaScript Design Patterns](#)
- [中文图解](#)

发布/订阅模式 OR 观察者模式

<https://www.cnblogs.com/lovesong/p/5272752.html>

练习

<https://github.com/FE-star/exercise17>

思考

1. 区别在哪?
2. 有什么应用场景?
3. 是否符合软件工程设计原则
4. 和接下来要讲的事件有什么关联

事件、事件模型、事件处理机制

从 addEventListener 说起

<https://developer.mozilla.org/zh-CN/docs/Web/API/EventTarget/addEventListener>

SHOWCASE

看看标准怎么说

1. <https://www.w3.org/TR/DOM-Level-3-Events/#dom-event-architecture>
2. <https://developer.mozilla.org/en-US/docs/Web/API/Event>

Event (事件)
被观察者

事件模型
观察者模式具
体实现

事件处理机制
事件传播的方式
捕获&冒泡

历史&兼容问题

```
<div onclick="alert(1)"></div>
```

```
element.onclick = function() {};
```

```
element.attachEvent('onclick', callback);
```

<https://caniuse.com/#search=EventTarget>

```
var EventUtil = {  
    // 添加事件监听  
    add: function(element, type, callback){  
        if(element.addEventListener){  
            element.addEventListener(type, callback, false);  
        } else if(element.attachEvent){  
            element.attachEvent('on' + type, callback);  
        } else {  
            element['on' + type] = callback;  
        }  
    }  
}
```

浏览器兼容

1. 渐进增强
2. 优雅降级

自定义事件

- 如何自定义事件? 参考
- 为什么要自定义事件?
- 应用场景有哪些?
 - 弹窗登录
 - toast

事件代理和委托

- <https://zhuanlan.zhihu.com/p/26536815>
- [\\$.delegate](#)
- [\\$.live](#)
- [\\$.on](#)
- 优缺点？
 - 减少内存占用
 - 一次委托、终身受用
 - 有一定的局限性

Events VS Event loop

- 不是同一个东西
- <https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/>
- <https://cnodejs.org/topic/57d68794cb6f605d360105bf>
- [http://menzhongxin.com/2017/05/18/node%E4%BA%8B%E4%BB%B6%E5%BE%AA%E7%8E%AF\(EventLoop\)/](http://menzhongxin.com/2017/05/18/node%E4%BA%8B%E4%BB%B6%E5%BE%AA%E7%8E%AF(EventLoop)/)

AJAX

Asynchronous JavaScript And XML

<https://developer.mozilla.org/zh-CN/docs/Web/Guide/AJAX>

ajax & jQuery.ajax

- ajax != jQuery.ajax
- ajax, 2005 年 2 月 18 日, Google
- Ajax: A New Approach to Web Applications
- jQuery, 2006 年 8 月 26 日, John Resig

xhr-XMLHttpRequest

```
var xhr = new XMLHttpRequest();

console.log('UNSENT', xhr.readyState); // readyState will be 0
xhr.open('GET', '/api', true);
console.log('OPENED', xhr.readyState); // readyState will be 1

xhr.onprogress = function () {
  console.log('LOADING', xhr.readyState); // readyState will be 3
};
xhr.onreadystatechange = function () {
  if(xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
    console.log(xhr.responseText);
  }
};
xhr.onload = function () {
  console.log('DONE', xhr.readyState); // readyState will be 4
};

xhr.send(null);
```

Fetch

```
const options = {
  headers: new Headers({ 'Content-Type': 'application/json' }),
  cache: 'default',
  method: 'GET',
  mode: 'cors',
};

function get(url, params) {
  const opts = Object.assign({}, options, { method: 'GET' });
  return fetch(`${url}?${querystring.stringify(params)}`, opts)
    .then(data => data.json());
}

function post(url, params) {
  const opts = Object.assign({}, options, { method: 'POST', body: JSON.stringify(params) });
  return fetch(url, opts)
    .then(data => data.json());
}
```

思考

有 xhr 为什么还要 fetch ?

异步编程

同步 VS 异步

sync

async

SHOWCASE

回调 (callback)

回调地狱 (callback hell)

可读性&可维护性

*Generator + function**

过渡阶段的产物

Promise + async/await

新的标准

RxJS

响应式编程

观察者模式+迭代器模式+Stream

Unix 设计思想--管道 & Stream

- <https://zh.wikipedia.org/wiki/Unix%E5%93%B2%E5%AD%A6>
- <https://coolshell.cn/articles/7236.html>
- <https://nodejs.org/api/stream.html>
- <https://medium.freecodecamp.org/node-js-streams-everything-you-need-to-know-c9141306be93>



IT前沿技术在线大学

www.zhenchuanx.com