

## 解决 es6 兼容 (编译 / 转换)

1. 在线转换 : 引入 js 库, 并声明类型 text/babel
2. 提前编译

## es6 新特性

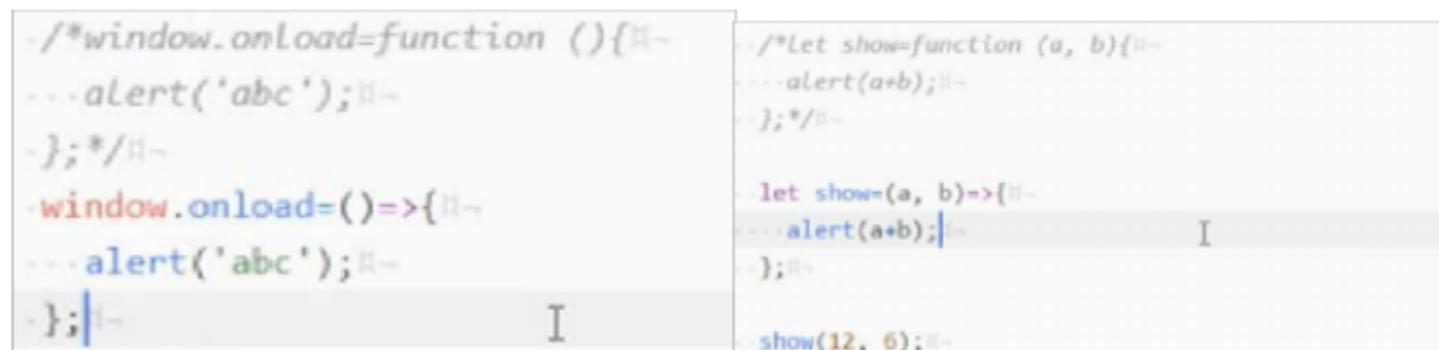
1. 变量
2. 函数
3. 数组
4. 字符串
5. 面向对象
6. Promise
7. Generater (异步)
8. 模块化

## 变量

1. var : 可重复声明 / 无限制修改 / 没有块级作用域
2. let : 不能重复声明 / 变量 / 可修改 / 有块级作用域
3. const : 不能重复声明 / 常量 / 不可修改 / 有块级作用域

## 函数 => 箭头函数

()=>{}



```

/*window.onload=function (){
  alert('abc');
};*/
window.onload={()=>{
  alert('abc');
}};

/*let show=function (a, b){
  alert(a+b);
};*/
let show=(a, b)->{
  alert(a+b);
};
show(12, 6);
  
```

1. 如果有一个参数 () 可以省略
2. 如果只有一个 return {} 可以省略

## 函数 - 参数

1. 参数扩展 / 数组展开

a. 收集剩余的参数

Function show(a,b,...args) \*args 自定义, 并且必须是最后一个

b. 展开数组

Let arr = [1,2,3]; alert(...arr);

效果 1, 2, 3 依次弹出 \*展开后效果和直接把数组内容写在这一样

2. 默认参数

a. 默认参数

Function show(a,b=5,c=9){...}

Show(99);

Show(99,9); \*传参直接覆盖, 不传参直接显示默认参数数值

## 结构赋值

1. 左右两边结构必须一样

a. Let [a,b,c] = [1,2,3]; // 数组

Console.log(a,b,c); 1 2 3

b. Let {a,b,c}= {a:1,b:2,c:3}; //json

Console.log(a,b,c); 1 2 3

c.

```
let [{a, b}, [n1, n2, n3], num, str]=[{a: 12, b: 5}, [12,5,8], 8, 'cxzcv'];
console.log(a,b,n1,n2,n3,num,str);
```

(1)

```
let [json, arr, num, str]=[{a: 12, b: 5}, [12,5,8], 8, 'cxzcv'];
console.log(json,arr,num,str);
```

(2)

2. 右边必须是个东西 (必须是个正规的数组, json, 变量。。。。)
3. 声明和赋值不能分开 (必须在一句话里完成)

数组

1. Map 映射：一个对一个

- a. [12,100,60]  
[不及格,及格,及格]

b.

```
let arr=[12,5,8];
let result=arr.map(function (item){
  return item*2;
});
```

=>箭头函数

```
let arr=[12,5,8];
let result=arr.map(item=>item*2);
alert(result);
```

结果：24, 10, 16

```
let score=[19, 85, 99, 25, 90];
let result=score.map(item=>item>=60?'及格':'不及格');
alert(score);
alert(result);
```

2. Redurt 汇总：一堆出来一个

- a. 算总和 Tmp：临时结果, item 下一个数字, index 当前位数

```
let arr=[12,69,180,8763];
let result=arr.reduce(function (tmp, item, index){
  //alert(tmp+', '+item+', '+index);
  return tmp+item;
});
alert(result);
```



b.

3. Filter 过滤器 例：可以整除 3 的选出来

```
let arr=[12,5,8,99,27,36,75,11];
let result=arr.filter(item=>item%3==0);
alert(result);
```

```
let arr=[
  {title: '男士衬衫', price: 75},
  {title: '女士包', price: 57842},
  {title: '男士包', price: 65},
  {title: '女士鞋', price: 27531}
];
let result=arr.filter(json=>json.price>=10000);
console.log(result);
```

#### 4. Foreach 循环/迭代

```
let arr=[12,5,8,9];
arr.forEach(item=>{
  alert(item);
});
```

#### 字符串

##### 1. 多了两个新方法

a. startsWith 例：str 字符串是否以 a 开始

```
let str = ' abcdefg ';
str.startsWith(' a');
```

b. endsWith 例：str 字符串是否以 .txt 结尾

```
let str = ' a.txt ';
Str.endsWith(' .txt');
```

##### 2. 字符串模板

a. let str = `abc`; 反双引号 ``

b. 拼接字符串换用

i. \${拼接的东西} // 可以直接把变量拼接进字符串

ii. 可以折行 / 可以换行

```
let a=12;
let str=`a${a}bc`;
```

#### es6 的面向对象

##### 封装

1. Class 关键字，构造器和类分开了

2. Class 里直接加方法

##### 继承

1. super--超类 / 父类

```
class User{
  constructor(name, pass){
    this.name=name;
    this.pass=pass;
  }
  showName(){
    alert(this.name);
  }
  showPass(){
    alert(this.pass);
  }
}
var u1=new User('blue', '123456');

class VipUser extends User{
  constructor(name, pass, level){
    super(name, pass);
    this.level=level;
  }
  showLevel(){
    alert(this.level);
  }
}
var v1=new VipUser('blue', '123456', 3);
```

#### 面向对象的应用 --React

React

1. 组件化 --class
2. JSX: js 扩展版

JSX==Babel==browser.js // 如果不写 " 长得像 html 的东西就默认为要创建一个 html

```
...window.onload=function (){...  
...let oDiv=document.getElementById('div1');...  
...ReactDOM.render(...  
...<span>123</span>,...  
...oDiv...  
...);...  
...};...
```

### Json

Json 标准格式：只能用双引号 / 所有名字必须用引号包起来

1. json 对象

- a. JSON.stringify() //json 对象转 json 字符串
- b. JSON.parse() //json 字符串转 json 对象

2. Json 简写：名字一样 / 方法

- a. 名字和值 ( key 和 value ) 一样的时候可以简写一个

```
...let a=12;...  
...let b=5;...  
...let json={a, b, c: 55};...
```

- b. Json里如果有方法，可以把： function 省略掉

```
...let json={...  
...a: 12,...  
...show(){...  
...alert(this.a);...  
...}...  
...};...
```

### Promise--承诺

// 消除异步操作：用同步的代码来书写异步代码

// 当状态改变的时候 --调用之前挂起的 then 队列

//then 的时候直接执行对应的函数，并且参数给人家

Promise.all: 一个一个都要成功

Promise.race：读到那个算哪个，可以不全部成功

1. 异步请求：操作之间没什么关系 / 同时进行多个操作

缺点：代码更复杂

2. 同步请求：同时只能做一件事 优点：代码更简单

如果两个 Ajax 请求则 then 是图二写法

```

let p=new Promise(function (resolve, reject){
  //异步代码
  //resolve--成功了
  //reject--失败了
  $.ajax({
    url: 'arr.txt',
    dataType: 'json',
    success(arr){
      resolve(arr);
    },
    error(err){
      reject(err);
    }
  });
  p.then(function (){
    alert('成功');
  }, function (){
    alert('失败了');
  });
});

Promise.all([
  p1, p2
]).then(function (){
  alert('全都成功了');
}, function (){
  alert('至少有一个失败了');
});

```

### 3. 简写

```

function createPromise(url){
  return new Promise(function (resolve, reject){
    $.ajax({
      url,
      dataType: 'json',
      success(arr){
        resolve(arr);
      },
      error(err){
        reject(err);
      }
    });
  });
}

Promise.all([
  createPromise('data/arr.txt'),
  createPromise('data/json.txt')
]).then(function (arr){
  let [res1, res2]=arr;
  alert('全都成功了');
  alert(res1);
  alert(res2);
}, function (){
  alert('至少有一个失败了');
});

```

### 4. 高版本 jQuery 已经封装了 Promise，可以直接用 Promise.all

```

Promise.all([
  $.ajax({url: 'data/arr.txt', dataType: 'json'}),
  $.ajax({url: 'data/json.txt', dataType: 'json'})
]).then(function (results){
  let [arr, json]=results;
  alert('成功了');
  console.log(arr, json);
}, function (){
  alert('失败了');
});

Promise.all([$ajax(), $ajax()]).then(results=>{
  //对了
}, err=>{
  //错了
});

```

### generator -生成器函数

普通函数 --开始运行函数中间不会停，知道运行结束为止

generator 函数 --中间可以停

// 比如打出租车，中途有事可以让司机停车，事办完了回来让司机继续走

1. \*...yield; //...t 停止运行 next(); //... 开始运行

```

function *show(){
  alert('a');

  yield;

  alert('b');
}

let genObj=show();

genObj.next();
genObj.next();

```

generator -yield 传参 / 返回

1. 传参

第一个 next 传参是没法传给 yield，此参数无用，第一部分过程正常传参就行

2. 返回

软件

```

软件:
1.编辑器: atom(插件)
2.服务器: wamp、xampp、nginx
  ·mysql、php
3.Photoshop CS6
4.手机虚拟机—夜神
5.Navicat for mysql
6.虚拟机VMware
7.centos镜像
  ·http://mirrors.163.com/centos/7.4.1708/isos/x86\_64/
  ·CentOS-7-x86_64-Minimal-1708.iso
8.PuTTY、winSCP(Mac用yummy FTP、Linux直接ssh)
9.git for windows (Mac用git-scm、Linux直接yum/apt)
10.TortoiseSVN (Mac用SnailSVN、Linux直接yum/apt)
11.Node.js
  ·https://nodejs.org/en/
12.phonegap-cli v6.5.2
  ·npm
13.APICloud Studio 2
14.MongoDB
  
```

webSocket : 数据交互

1. 性能高

2. Websocket 基于 http 的 socket 建立连接的部分，连接建立之后，就会变成二进制的连接

3. 双向 -数据实时性高

4. Html --ie9+

5. Socket.io

a. 兼容

b. 二进制

计算机网络的组成 socket == Tcp == node里的 net

OSI 7层参考交换模型		
物理层	编码、材质、造价、电压	CDMA
链路层	内网传输 内网寻址	arp
网络层	外网传输 外网路由	IP
传输层	传输质量 可靠连接(保证到达、保证正确、保证顺序)	TCP
表现层	屏蔽不同网络类型之间的差异	x
会话层	保持双方的状态	x
应用层	具体应用相关的功能	http/ftp/pop3/smtp/.../webSocket

## Websocket

Web socket ws:// 普通连接

Web socket security wss:// 加密连接

```

1 const net = require('net');
2 //1. 创建一个tcp服务器
3 let server = net.createServer(socket=>{
4   console.log('有人连接我了!');
5   //3. 接收浏览器发过来的请求头, 处理, 返回数据给浏览器
6   socket.on('data', data=>{
7     //第一步
8     //第一步, 把数据转换成headers的json
9     let str = data.toString();
10    let aHeaders = str.split('\n\n');
11    aHeaders.shift();
12    aHeaders.pop();
13    aHeaders.pop();
14    let headers = {};
15    aHeaders.forEach(str=>{
16      let [name,value] = str.split(': ');
17      headers[name] = value;
18    });
19    console.log(headers);
20    //第二步, 校验
21    if (headers['Connection']!=='Upgrade' || headers['Upgrade']!=='websocket') {
22      console.log('收到了一个ws以外的协议');
23      socket.end();
24    }else {
25      //第三步, 校验websocket
26      if (headers['Sec-WebSocket-Version']!=13) {
27        console.log('出现了意外的ws版本');
28        socket.end();
29      }else {
30        //第四步, 生成key
31        let hash = crypto.createHash('sha1');
32        hash.update(headers['Sec-WebSocket-Key']+'258EAFA5-E914-47DA-95CA-CSAB8DCB5B11');
33        let base64Key = hash.digest('base64');
34        socket.write('HTTP/1.1 101 Switching Protocols\r\nUpgrade: websocket\r\nConnection: Upgrade\r\nSec-WebSocket-Accept: ${base64Key}\r\n\r\n');
35        console.log('握手结束了');
36      }
37    }
38  });
39  socket.on('end', ()=>{
40    console.log('断开连接了');
41  });
42 });
43 server.listen(8080);

```

```

<script type="text/javascript">
  //2. 连接到服务器
  let ws = new WebSocket('ws://localhost:8080/');
  ws.onopen = function(){
    alert('连接已打开');
  }
  ws.onmessage = function(){
    alert('有消息过来');
  }
  ws.onclose = function(){
    alert('连接已关闭');
  }
  ws.onerror = function(){
    alert('连接出错');
  }
</script>

```

1. Qw
- 2.

后台 node :

1. 原生 nodejs:javascript 后台 (小型项目, 工具型项目 / 中间层语言 )
  - a. 性能高
  - b. 和前台配合方便 / 适合前端人员入门

- c. 协议
  - I. listen---等待客户连接
  - II. 端口 --数字：区分不同服务（没被占用的， linux1024+ 以上的端口）
- d. 每种服务都有默认的端口
  - I. web--80
  - II. Ssh--22
  - III. ftp--21
  - IV. Mysql--3306
  - V.
- e.

```

const http = require('http');
let server = http.createServer(function(){
    console.log('有人请求我, 并且success');
});
server.listen(8080);
console.log('success');
    
```

```

E:\>cd node
E:\node>node server.js
SUCCESS
有人请求我, 并且success
    
```

```

const http = require('http');
let server = http.createServer(function(req,res){
    //request-----请求-输入-请求信息, 哪个地址, 时间, IP, 方法, ...
    //response-----响应-输出
    console.log('有人请求我, 并且success');
    console.log('客户端请求的是: ${req.url}');
    console.log('请求的方法是: ${req.method}');
    res.write('好的我知道了!');
    res.end();
});
server.listen(8080);
console.log('success');
    
```

```

E:\>cd node
E:\node>node server.js
SUCCESS
有人请求我, 并且success
客户端请求的是: /index.html
请求的方法是: GET
    
```

g. 1

h.

## 2. Nodejs 框架

### Html5 新特性

#### 1. geolocation--定位

- a. PC--ip 地址（精度比较低、 ip 库）
- b. 手机 ---GPS( )
  - I. 单次： window.navigator.geolocation( 成功, 失败, 参数 )
    - enableHighAccuracy 高精度模式 ---更慢、更费电
    - Timeout 超时
    - MaximumAge 缓存时间
  - II. 结果： latitude/longitude 维度 /精度
    - Altitude 海拔高度
    - Accuracy 精确度

AltitudeAccuracy	高度精确度
Heading	朝向
Speed	速度

### III. 监听：watchPoition（成功，失败，参数）

```

<script type="text/javascript">
window.onload = function(){
let btn1 = document.getElementById('btn1');
btn1.onclick = function(){
if (window.navigator.geolocation) {
navigator.geolocation.getCurrentPosition(res=>{
alert('success')
},err=>{
alert('error')
},{
enableHighAccuracy // 高精度模式
});
}else {
alert('浏览器不支持定位！')
}
}
}
</script>

```

## 2. Video、audio

### a. video---视频

```

<video>
--<flash></flash>
</video>

```

- I. src 地址
- II. autoplay 自动播放
- III. loop 循环播放
- IV. poster 封面地址
- V. controls 显示播放器的控件
- VI. Video 支持什么格式 ---通用格式 MP4
  - IE wmv、MP4
  - Chrome webq、MP4
  - FireFox ogv、MP4

### VII.

### b. Audio---音频 MP3

### c. Js接口

- I. .play() 播放
- II. .pause() 暂停
- III. .stop() X
- IV. .currenttimr() 当前播放的位置 ( s )
- V. .duration() 长度 ( s )
- VI. .volume() 音量 0-100
- VII. .muted() 静音： bool

### d.

- 3. LocalStorage
- 4. Websql/indexedDB
- 5. Webworker --多线程 (提高用户体验，提高性能)

主机 > 程序 > 进程 > 线程 > 纤程

创建子进程 -> 发送数据 -> 接收数据 -> 处理数据 -> 返回结果 -> 接受结果

	多进程（重）	多线程（轻）
开销	创建、销毁开销大	创建、销毁开销小
安全性	进程之间是隔离	线程之间是共享
资源	每个进程独立资源	同一个进程的所有线程共享资源
共享资源	麻烦	方便
编程难度	低（资源是独享的）	高（资源是共享的）

```

<script type="text/javascript">
window.onload = function(){
let btn = document.getElementById("btn1");
let txt1 = document.getElementById("txt1");
let txt2 = document.getElementById("txt2");
btn.onclick = function(){
let n1 = parseInt(txt1.value);
let n2 = parseInt(txt2.value);
//1. 创建子进程
let w = new Worker("w1.js");
//2. 发送数据
w.postMessage({n1,n2});
//3. 接收结果
w.onmessage = function(ev){
console.log(ev.data);
}
};
}
</script>

```

```

//3. 接收数据
this.onmessage = function(ev){
console.log(data);
//4. 处理数据
let sum = ev.data.n1 + ev.data.n2;
//5. 返回结果
this.postMessage(sum);
}

```

- a. 优点：
  - I. 充分利用资源（多个进程同时工作）
  - II. 防止主进程卡住
- b. 缺点：
  - I. 不能执行任何 ui 操作，子进程只能执行计算型任务
- c.

6. 文件操作 / 拖拽

7. Manifest

8. Canvas-画布（什么都能画，性能高，只能用宽高属性修改大小，，只要画完了就不能修改）

a. 路径操作

- moveTo/lineTo 开始 / 结束
- .beginPath() ---清除
- .closePath() ---闭合
- .rect() -----画矩形
- .arc(cx,cy,r startAng,endAng, 逆时针 ) -----画圆

b. 非路径

- .strokeRect()
- .fillRect()

c. 画线，填充，颜色 // 属性

.stroke()  
.strokeStyle()  
.fill()  
.fillStyle()

d.

画线条

```
<script type="text/javascript">
  let oc = document.getElementById('canvas');
  let gb = oc.getContext('2d'); // 图形上下文, 绘图用的接口
  gb.moveTo(100,100);
  gb.lineTo(300,200); // 只是范围不会出现图形
  gb.stroke(); // 线条
  gb.fill(); // 填充
</script>
```

```
let l=50, t=50, w=100, h=70;
//矩形
gd.strokeRect(l, t, w, h);

oC.onmousemove=function (ev){
  gd.clearRect(0,0,oC.width,oC.height);

  if(
    ev.offsetX>=l &&
    ev.offsetX<=l+w &&
    ev.offsetY>=t &&
    ev.offsetY<=t+h
  ){
    gd.strokeStyle='red';
    gd.strokeRect(l, t, w, h);
  }else{
    gd.strokeStyle='black';
    gd.strokeRect(l, t, w, h);
  }
};
```

### 9. 拖拽

- reader.readAsText(ofile)      文本
  - reader.readAsDataURL(ofile)    读取图片 --base64
  - reader.readAsArrayBuffer(ofile)    二进制数据，他是一个数组
  - reader.readAsBinaryString(ofile)    二进制数据，会把内容转成字符串
- JS高级语言，不擅长处理二进制

```
<script type="text/javascript">
  window.onload = function(){
    let oDiv = document.getElementById('div1');
    oDiv.addEventListener('dragenter',function(){
      oDiv.innerHTML = '请放手';
    },false);
    oDiv.addEventListener('dragleave',function(){
      oDiv.innerHTML = '推到这里上传';
    },false);
    oDiv.addEventListener('dragover',function(ev){
      ev.preventDefault();
    },false);
    oDiv.addEventListener('drop',function(ev){
      ev.preventDefault();
      let ofile = ev.dataTransfer.files[0];
      let reader = new FileReader();
      reader.onload = function(){alert('读取成功'+this.result)}; // this 指的是reader
      reader.onerror = function(){alert('读取失败')};
      reader.readAsText(ofile);
    },false);
  }
</script>
```

reader.readAsText(ofile)

- I. ondragenter 进去
  - II. ondragleave 出来
  - III. ondragover 悬停 --一直发生
  - IV. ondrop 松手
- 10.

## 1. 1.移动端

### a. 布局

#### I. Viewport (视口) 调整浏览器分辨率

Content= '手机推荐尺寸, 是否可缩放, 初始默认的缩放比例 1.0, 最大可缩放 2倍'

```
<meta name="viewport" content="width=device-width,user-scalable=0,initial-scale=1.0,max-scale=2.0">
```

#### II. 弹性盒模型 宽/高/padding/maigin

父级 ----display:flex; // 让父级元素成为一个盒模型

子级 ----flex : 1 // 让子元素自动按比例分大小

可用总宽 -固定总宽 =剩余宽度

剩余空间 \*flex/flex\_sum=width

```
.ul1 {display:flex;}
.ul1 li {flex:1; background:#CCC; border:1px solid black;}
```

#### III. 新的盒模型

box-sizing :

Content-box 宽高指的是内容宽高

Border-box 宽高指的是 border 外面

```
box-sizing: border-box;
```

百分比漂浮的话, 家 border 防止换行

#### IV. 响应式布局 (响应式设计) 一套页面可以应用所有平台 (pc端, pad端, 手机端) // 适用小网站

- I. 媒体查询 @media// 类似 if 小于 400 像素执行红色, 超过 400 执行绿色  
有优先级

```
.box {height:30px; background:green;}
@media (max-width:400px) {
  .box {background:red;}
}
```

#### V. Rem 单位

- I. Rem 相对于根元素 (html) 的字体大小

给一个 rem 基准 (就是给 html 一个 font-size 值)

Rem 换算: 真实屏幕 / rem 基准

```

<meta name="viewport" content="width=device-width,user-scalable=no,initial-scale=1.0">
<title></title>
<style media="screen">
  *{margin:0;padding:0;list-style:none;}
  html{font-size:20px}
  .banner{}
  .banner li{width:18.75rem;height:5.954255rem;}
  .banner li img{width:100%;height:100%;}
</style>
<script type="text/javascript">
  document.documentElement.style.fontSize = 20*document.documentElement.clientWidth/375+'px';
</script>
</head>
<body>
<ul class="banner">
  <li>
    
  </li>
</ul>

```

II. Em 相对字体的大小 font-size: 12px;width:2em => width : 24px

VI.

b. Touch--多点触摸、手势、方向锁定、页面滚动 ---iscroll、hammer

I. ontouchstart onmousedown

II. ontouchmove onmousemove

III. ontouchend onmouseup

多点/单点

```

//drag
<meta charset="utf-8">
<meta name="viewport" content="width=device-width,user-scalable=no,initial-scale=1.0">
<title></title>
<style media="screen">
  .box{width:100px;height:100px;background:#ccc;position:absolute;}
  .a{background:red;left:50px;top:50px;}
  .b{background:blue;left:110px;top:50px;}
  .c{background:yellow;left:150px;top:150px;}
</style>
<script type="text/javascript">
window.onload = function(){
  let abox = document.getElementsByClassName('box');
  Array.from(abox).forEach(box=>{
    drag(box);
  });
  function drag(obj){
    obj.addEventListener('touchstart',ev=>{
      let disx = ev.targetTouches[0].pageX-obj.offsetLeft;
      let disy = ev.targetTouches[0].pageY-obj.offsetTop;
      ev.cancelBubble = true;
      ev.preventDefault();
      function fmove(ev){
        obj.style.left = ev.targetTouches[0].pageX-disx+'px';
        obj.style.top = ev.targetTouches[0].pageY-disy+'px';
        ev.preventDefault();
      }
      function fend(){
        obj.removeEventListener('touchmove',fmove,false);
        obj.removeEventListener('touchend',fend,false);
        obj.addEventListener('touchmove',fmove,false);
        obj.addEventListener('touchend',fend,false);
      },false);
    }
  }
</script>
</head>
<body>
<div class="box a"></div>
<div class="box b"></div>
<div class="box c"></div>
</body>
</html>

<meta name="viewport" content="width=device-width,user-scalable=no,initial-scale=1.0">
<title></title>
<style media="screen">
  .box{width:100px;height:100px;background:#ccc;position:absolute;left:0;top:0;}
</style>
<script type="text/javascript">
window.onload = function(){
  let abox = document.getElementsByClassName('box')[0];
  abox.addEventListener('touchstart',function(ev){
    //开始
    let disx = ev.targetTouches[0].pageX-obj.offsetLeft;
    let disy = ev.targetTouches[0].pageY-obj.offsetTop;
    ev.cancelBubble = true;
    ev.preventDefault();
    function fmove(ev){
      abox.style.left = ev.targetTouches[0].pageX-disx+'px';
      abox.style.top = ev.targetTouches[0].pageY-disy+'px';
      ev.preventDefault();
    }
    function fend(){
      abox.removeEventListener('touchmove',fmove,false);
      abox.removeEventListener('touchend',fend,false);
      abox.addEventListener('touchmove',fmove,false);
      abox.addEventListener('touchend',fend,false);
    },false);
  });
}
</script>

```

IV.

c.

2. 图形

a. canvas 位图，方法大会失真，不存储图形信息； ----- 没法修改 / 没有的事件；性能高

b. Svg/vml 矢量图，放大不会失真，存储图形信息； ----- 便于修改，事件，性能一

3. 般