

# HTML 5

---

- (1) 新语义标签
- (2) 增强型的表单
- (3) Canvas 绘图 -- ( 重难点 )
- (4) 音频和视频 -- ( 视频 : 重点 )
- (5) SVG绘图
- (6) 拖动 API
- (7) 地理定位
- (8) Web Worker
- (9) Web Storage
- (10) Web Socket

## 增强型的表单

---

- **新 input type**

h4 : text; password; checkbox; button; submit; reset; radio

h5 : email; number; search; color; date; month; week

- **新 element**

h4 : input; textarea; select / option; label;

h5 : datalist; progress; meter; output

- **新 attr**

h4 : id / class / title / type / name / disabled / readonly ...

h5 : placeholder 占位符

### 1.1 datalist 建议列表

---

建议列表

```
1 <datalist id="list3"> // datalist 本身不可见
2   <option>xx</option>
3   <option>yy</option>
4 </datalist>
5 <input type="text" list="list3">
6 // datalist 为 input 提供一个输入建议，用户可以从选取，也可以手工输入
```

## 1.2 progress 进度条

显示一个进度条，有两种形式

`<progress>` 左右晃动进度条

`<progress value="0.7">` 具有指定的进度值的进度条

提示：项目中，推荐 canvas 来代替 progress

## 1.3 meter 刻度尺

用于标识一个值所处范围：红色（不可接受）黄色（可接受的）绿色（非常合理的）

```
1 <meter
2   min="最小值" max="最大值"
3   low="合理下限值" high="合理上限值"
4   optimum="最佳值"
5   value="当前值">
6 </meter>
```

## 1.4 output

output：输出；语义标签，没有任何外观样式，样式上等同于 span

EX:

商品单价：¥3.50

购买数量：`<input type="number" value="1">`

小计：`<output> ¥3.50</output>`

## 1.5 新属性

**placeholder** 占位符

`<input type="text" placeholder="请输入用户名">`

**autofocus** 自动获取输入焦点

`<input autofocus>`

**multiple** 允许输入框中出现多个输入值（用逗号隔开）

ex : 邮箱输入

**form** 用于把输入域放置到 form 外部

```
<form id="f5"></form>
```

输入验证 -- **required** 必填项，内容不能为空

输入验证 -- **maxlength** 指定字符串最大长度

输入验证 -- **minlength** 指定字符串最小长度

输入验证 -- **min** 指定数字的最小值

输入验证 -- **max** 指定数字的最大值

输入验证 -- **pattern** 正则表达式

## 视音频（重点）

---

### (1) Flash

播放视频音频通常技术

flash 绘图	=> Canvas / SVG
flash 动画	=> Canvas + 定时器
flash 视音频	=> video / audio
flash 客户存储	=> WebStorage

---

### (2) H5视频

H5 提供了一个新标签用于播放视频

```
<video src="res/birds.mp4"></video>
```

浏览器内部解码器不足？

解决方案一：

格式工厂 x.mp4 -> x.avi -> x.ogg -> x.webm

```
<video>
  <source src="x.mp4">
  <source src="x.webm">
  <source src="x.ogg">
</video>
```

---

成员属性	默认值	描述
autoplay	false	是否自动播放
controls	false	是否显示控件
loop	false	是否循环播放
muted	false	是否静音
poster		在播放第一帧之前的海报
preload	metadata	视频预加载策略
	auto	预加载视频元数据及缓冲一定时长视频
	metadata	仅预加载元数据（时长；尺寸；第一帧）
	none	不预加载任何数据

js 对象属性	默认值	描述
paused	true	当前视频是否处于暂停状态
volume	1	当前音量 0 - 1
playbackRate	1	播放速度

js 对象成员方法	描述
play()	播放视频
pause()	暂停视频

js 对象成员事件	描述
onplay	当视频开始播放时触发的事件
onpause	当视频暂停播放时触发的事件

### (3) H5音频

H5 提供了一个新的标签用于播放音频

```
<audio src="x.mp3"></audio>

<audio>
  <source src="x.mp3">
  <source src="x.ogg">
</audio>
```

默认是一个300\*30的inline-block元素,但若没有controls属性

属性	默认值	描述
autoplay	false	是否自动播放
controls	false	是否显示控件
loop	false	是否循环播放
muted	false	是否静音
preload	metadata	视频预加载策略
	auto	预加载视频元数据及缓冲一定时长视频
	metadata	仅预加载元数据（时长；尺寸；第一帧）
	none	不预加载任何数据

js 对象属性	默认值	描述
paused	true	当前音频是否处于暂停状态
volume	1	当前音量 0 - 1
playbackRate	1	播放速度

js 对象成员事件	描述
onplay	当音频开始播放时触发的事件
onpause	当音频暂停播放时触发的事件

## H5 Canvas

网页中实时走势图，统计图，网页游戏，地图应用

三种绘图技术：

- (1) Canvas 绘图：2D 位图绘图技术，H5 推出绘图技术
- (2) SVG 绘图：2D 矢量图绘图技术，2000年出现后纳入 H5 标准
- (3) WebGL 绘图：3D 绘图技术，尚未纳入 H5 标准

Canvas 绘图难点：

- (1) 坐标系
- (2) 单词比较多

先创建画布

```
<canvas width="500" height="400">
```

您的浏览器版本过低，请更新！

```
</canvas>
```

canvas 标签在浏览器中默认 300\*150 的 inline-block

**注意：画布宽度高度只能使用 js / html 属性来赋值，不能使用 css 样式来赋值**

每个画布上有且只有一个“画笔”对象

```
var ctx = canvas.getContext("2d");
```



## Canvas 矩形

矩形的定位点在自己的左上角

属性	属性值	描述
ctx.lineWidth	1	描边宽度
ctx.fillStyle	"#000"	填充样式
ctx.strokeStyle	"#000"	描边样式
方法		
ctx.fillRect(x,y,w,h);		绘制填充的矩形
ctx.strokeRect(x,y,w,h);		绘制描边的矩形
ctx.clearRect(x,y,w,h);		清除一个矩形范围内的所有绘图

## Canvas 文本

属性	属性值	描述
ctx.textBaseline	top	文本基线
ctx.font	12px sans-serif	文本大小和字体
方法		
ctx.fillText(str,x,y)		填充一段文本
ctx.strokeText(str,x,y)		描边一段文本
ctx.measureText(str).width		测量一段文本宽度，返回一个对象{width:"x"}

## Canvas 渐变

渐变可以填充在矩形, 圆形, 线条, 文本等等, 各种形状可以自己定义不同的颜色

### 线性渐变

(1) 创建渐变对象 ( 起点坐标 / 终点坐标 )

```
var g = ctx.createLinearGradient(x1,y1,x2,y2);
```

(2) 添加颜色点

```
g.addColorStop(offset,color);
```

(3) 将渐变对象赋值给描边或者填充样式

```
ctx.strokeStyle = g;
```

```
ctx.fillStyle = g;
```

## Canvas 路径

path : 由多个坐标点组成的任意形状，路径不可见，可用于“描边”，“填充”

方法	取值	描述
ctx.beginPath();		开始一条新路径
ctx.closePath();		闭合当前路径
ctx.moveTo(x,y);		移动到指定点
ctx.lineTo(x,y);		从当前点到指定点绘制直线
ctx.arc(cx,cy,r,start,end);		绘制圆拱形
	cx,cy	原点
	r	半径
	start	起始角度 0 ~ 360
	end	结束角度 0 ~ 360
ctx.stroke();		描边
ctx.fill();		填充

## Canvas 变形

canvas 属于客户端技术，  
图片在服务器中，所有浏览器必须先下载要绘制的图片，  
且等待图片异步加载成功后绘制

图片步骤：

1. 创建图片对象 `var p3 = new Image();`
2. 下载图片 `p3.src = "x.png";`
3. 绑定事件 `p3.onload = function(){} 图片加载成功`
4. 绘制图片 `ctx.drawImage(p3,x,y);` 原始大小  
`ctx.drawImage(p3,x,y,w,h);` 拉伸绘图

变形步骤



方法	描述
ctx.rotate(deg);	旋转画笔对象，轴点为画布的原点
ctx.translate(x,y);	将整个画布的原点平移到指定的点
ctx.save();	保存画笔当前所有的状态（角度/原点/颜色...）
ctx.restore();	恢复画笔状态到最近一次保存中的状态

提示：同一个画布上绘制多个角色 = 套路

以前：平移 + 旋转 + 逆向旋转 + 逆向平移

现在：保存 + 平移 + 旋转 + 绘制 + 恢复画笔

## SVG

**位图**：由一个个像素组成，每个点各有自己的颜色 => 颜色细腻，但是放大后失真

**矢量图**：由一个个线条组成，每个线条指定颜色，方向，可以无限缩放，但是细节不够丰富

svg (可缩放的矢量图)

	Canvas	SVG
类型	2D 位图	2D 矢量图
如何绘图	js 代码绘图	标签绘图
事件绑定	每个图形不是元素， 无法直接绑定事件 只能绑定画布	每个图形都是元素 可以直接绑定事件
应用场合	游戏，特效	地图

SVG 技术诞生于 2000 年，早期作为 XML 的扩展，H5 把常用 SVG 标签采纳为标准，但有一些废弃

## svg 矩形

```

1  <svg id="" width="" height="">
2    svg 画布
3    <rect width=""
4          height=""
5          x="" y=""
6          fill="" stroke=""
7          fill-opacity="" stroke-opacity="">
8    </rect>
9  </svg>

```

绘图特点：

- (1) 所有图形默认只有填充色（黑色）没有描边色
- (2) svg 图形的样式可以用元素属性方式声明，也可以使用css样式声明，只能使用svg专用样式
- (3) 图形可以使用JS 来对属性赋值，但不能用 html dom，只能用 核心DOM

```
r.x=10;r.width=100; //无效
```

```
r.setAttribute("x",10);r.setAttribute("width",100); //有效
```

- (4) 动态添加 svg 图形

#html 字符串拼接

```
var html = "<rect></rect>";
```

```
svg.innertHTML = html;
```

#dom 元素创建

```
var rect = document.createElementNS("http://www.w3.org/2000/svg", "标签名");
```

```
svg.appendChild(rect);
```

## svg 圆形

```
1 <circle r="" cx="" cy="" fill="" stroke=""></circle>
```

## svg 椭圆

```
1 <ellipse rx="" ry="" cx="" cy=""></ellipse>  
2 // rx:水平半径 ry:垂直半径
```

## svg 直线

```
1 <line x1="" y1="" x2="" y2=""></line>
```

## svg 折线

```
1 // 一条折线上可以有任意多个连续的点  
2 <polyline points="50,50 100,50 80,80 .."></polline>
```

## svg 文本

svg 画布上不允许使用普通的 HTML 绘制文本" 如 p,spanideo

```
1 <text font-size="" fill="" stroke="" x="" y="">文本的内容</text>
```

## svg 图像

```
1 <image xlink:href="x.png" ></image>
```

## svg 渐变对象

```
1 <defs> // 定义特效对象:渐变对象属于一种特效对象
2   <linearGradient id="g3" x1="" y1="" x2="" y2="">
3     <stop offset="0" stop-color="red"/>
4   </linearGradient>
5 </defs>
6 <ANY fill="url(#g3)" stroke="url(#g3)"></ANY>
```

## svg 滤镜

```
1 <defs>
2   <filter id="f2">
3     <feGaussianBlur stdDeviation="3" />
4   </filter>
5 </defs>
6 <ANY filter="url(#f2)"></ANY>
```

## H5 地理定位

---

**Geolocation** : 地理定位,使用 js 获取当前浏览器所在地理坐标

(经度/纬度/海拔/速度)数据,用于实现 LBS应用

(Location Based Service) 如饿了么,高德导航,滴滴打车...

手机浏览器如何获取定位信息

(1) 首先手机中GPS芯片与卫星通信,定位精度在米

(2) 然后手机通信基站定位获取,定位精度在公里

PC 浏览器如何获取定位信息

通过IP地行反向解析

**国内二家定位服务提供商**

(1) 百度地图

(2) 腾讯地图

html 中提供一个新对象,用于获取当前浏览器定位信息

window.navigator.geolocation{

```

getCurrentPosition : fn // 获取当前定位信息

watchPosition : fn // 监视定位数据变化

clearWatch : fn // 取消监视
}

// 需要链接到Google服务

```

## 百度地图

(1) 注册百度开发者帐号

<https://lbsyun.baidu.com/> 手机



(2) 创建一个网站,登录百度地图,为网站申请一个地图的AccessKey

(3) 在自己网页中嵌入百度地提供API

var map = new BMap.Map("container"); 创建地图实例

var point = new BMap.Point(116,39); 创建坐标点

map.centerAndZoom(point,17); 指定中心, 缩放比例

工具 --> 拾取坐标系统

## 拖放API

Drag & Drop :拖动和释放

拖放的源对象	描述
dragstart	拖动开始
drag	拖动中
dragend	拖动结束

拖放的目标对象	
dragenter	拖动进入
dragover	拖动悬停
dragleave	拖动离开
drop	释放

注意：必须组织 dragover 的默认事件，否则 drop 不能触发

整体拖动过程:  $\text{dragenter} * 1 + \text{dragover} * n + \text{dragleave} * 1$

整体拖动过程:  $\text{dragenter} * 1 + \text{dragover} * n + \text{drop} * 1$

## Web Worker

程序：Program 指可被 CPU 执行的代码，储存在外存（磁盘）中

进程：Program / Task 将程序调入内存，分配执行空间，随时供 CPU 调用执行

线程：Thread 线程是进程内部代码基本单位

### 进程 和 线程

- (1) 进程是操作系统分配内存的基本单位
- (2) 线程是 cpu 执行代码的基本单位
- (3) 线程必须处于某个进程内部
- (4) 一个进程必须有至少一个线程
- (5) 一个操作系统中可能同时存在几千个线程，它们都是“**并发**执行的”

宏观上看是同时执行，微观上看依然是依次执行的

### Chrome 浏览器中线程模型

一个 Chrome 浏览器进程中，至少有 6 个线程，

**资源请求线程**：可以 **并发** 的向 web 服务器 发起 http 请求，以获得所需资源

**UI 主线程**：负责所有内容的绘制到浏览器页面，还要负责执行 js 程序

```
1 <button>按钮一</button>
2 <script src="x.js"></script>
3 <button>按钮二</button>
4 // 如果 x.js 载入很慢，导致 按钮二 会在一段时间后才会出现
5 // 浏览器执行 js 只能渲染网页同一个线程 -- UI主线程
```

解决方案：创建一个并发执行新线程，让它执行耗时 js 任务

---

## H5 Web Worker -- 缺点

浏览器不允许 Worker 线程操作任何 DOM/BOM 对象

原因：浏览器只允许 UI 主线程操作 DOM/BOM，若多个线程同时运行，可能造成页面混乱

---

## H5 Web Worker -- 发送和接收数据

Worker 线程发送数据给 UI 数据

(1) UI 接受数据

```
var w = new Worker("js/03.js");
```

```
w.onmessage = function(e){e.data};
```

(2) Worker 发送数据

```
postMessage("str");
```

UI 线程发送数据给 Worker 数据

(1) Worker 接受数据

```
onmessage = function(e){e.data}
```

(2) UI 发送数据

```
var w = new Worker("js/03.js");
```

```
w.postMessage("str");
```

---

## H5 Web Storage

在浏览器中存储当前用户专用数据：购物车；定制样式

在客户端存储数据可以使用技术

(1) Cookie 技术：**浏览器兼容性好**，不能超过 4k，操作复杂

(2) Flash 存储：依赖于 Flash 播放器

(3) **H5 WebStorage**：不能超过 8M，操作简单

(4) IndexedDB：可以存储大量数据，还不是标准技术

强调：不要将 **安全性级别高** 的数据保存在客户端（密码）

---

Session：会话：浏览器从打开某个网站的第一个网页开始，中间打开多个网页，直到关闭浏览器的整个过程

---

WebStorage 为客户提供两个对象

(1) **sessionStorage** 类数组对象，会话级别的存储

描述：在浏览器的进程所分配的内存中，存储该会话中使用到的数据，仅供此次会话中所有程序使用，一旦浏览器关闭，数据清空

作用：在同一个会话中所有页面共享此数据（登录用户编号，昵称 ...）

**保存数据：**

```
sessionStorage[key] = val;  
sessionStorage.setItem(key,val);
```

**获取数据：**

```
var val = sessionStorage[key];  
var val = sessionStorage.getItem[key];  
var key = sessionStorage.key(i); 获取第i个数据  
sessionStorage.length;    数据个数
```

**删除数据：**

```
sessionStorage.removeItem(key);  删除指定数据  
sessionStorage.clear();    删除所有数据
```

## (2) localStorage(2) localStorage 类数组对象，跨会话级别的存储

**保存数据：**

```
localStorage[key] = val;  
localStorage.setItem(key,val);
```

**获取数据：**

```
var val = localStorage[key];  
var val = localStorage.getItem(key);  
var key = localStorage.key(i);  获取第 i 个数据  
localStorage.length;    数据个数
```

**删除数据：**

```
localStorage.removeItem(key); 删除指定数据  
localStorage.clear();    删除所有数据
```

---

localStorage 中若数据发生变化，会触发事件 window.onstorage，可以监听此事件，实现监听 localStorage 数据改变的要求，不能监听 sessionStorage

# WebSocket

---

代码不复杂原理重要

---

- HTTP 协议
    - 属于“请求-响应”模型，只有客户端发起请求消息，服务器才会返回响应消息
    - 没有请求就没有响应，一个请求只能得到一个响应
    - 有些场景中，此模型力不从心，比如股票实时走势图
    - 解决方案：  
长轮询 / 心跳请求 -- 定时器 + ajax  
请求过于频繁，服务器压力太大，不够频繁，用户数据延迟较大
- 

- WebSocket 协议
  - 属于“广播-收听”模型  
客户端连接服务器，不再断开，**永久连接**，双方就随时向对方发送消息

创建服务器：php / java / node.js

创建客户端：H5 创建新对象 WebSocket

创建客户端

- 连接到 ws 服务器
  - `var s = new WebSocket("ws://127.0.0.1:9001");`
- 向服务器发送数据
  - `s.send("str");`
- 接受服务器发来的消息
  - `s.onmessage = function(e){e.data};`
- 断开连接
  - `s.close();`