HTML5　Day01

```
<title>INPUT 新类型练习</title>
<script>
	/*需求
	  * 3 个滑动条表示颜色中的三原色
	  *  每次颜色的范围  - 0 ~ 255
	  *  设置 div 的背景颜色中(红、绿和蓝的值)*/
	// 1. 获取 3 个 input 元素
var inputs = document.getElementsByTagName("input");
	// 2. 遍历 3 个 input 元素,得到每个 input 元素
	for(var i=0;i<inputs.length;i++){
		var input = inputs[i];
		// 3. 为每个 input 元素绑定 onchange 事件
		input.onchange = mychange;
	}
	// 数组用于存储三个颜色值
	var arr = [];
	// 定义事件的处理函数
	function mychange(){
		// a. 获取 3 个 input 的 value 属性值
		arr[0] = inputs[0].value;
		arr[1] = inputs[1].value;
		arr[2] = inputs[2].value;
		// b. 设置 div 元素的背景颜色   rgb(0,0,0)
var div = document.getElementById("showcolor");
		div.style.background                                    =
"rgb("+arr[0]+","+arr[1]+","+arr[2]+")";
	}
  </script>
```

<title>HTML5 新元素</title>

```
	// 实现动态进度条的效果
	(function fn1(){
		var progress = document.getElementById("progress");
		var max = progress.max;
		var value = progress.value;
		if(value == max){
			clearTimeout(t);
		}
		value++;// 将 value 值增加
		// 将增加后的 value 值,重新设置 value 属性
		progress.value = value;
		t = setTimeout(fn1,100);// 设置定时器
	})();
	// 实现刻度的动态效果(从 min 自增 max)
	(function fn2(){
		var meter = document.getElementById("meter");
		var max = meter.max;
		var value = meter.value;
		if(value == max){
			cleatTimeout(t);
		}
		value++;
		meter.value = value;
		t = setTimeout(fn2,100);
	})();
```

<title>HTML5 的有效状态</title>

```
	/*使用 HTML5 中的验证属性时
	  * required 为 input 元素绑定事件(?)
	    *  底层具有阻止事件冒泡
	  *  为 form 表单绑定 onsubmit 事件
	    * onsubmit 事件导致失效  */
	function formValidate(){
		// 获取 input 元素
		var input1 = document.getElementById("i1");
		// 判断 input 元素是否为空
		if(input1.validity.valid){
			console.log("input 元素不为空.");
		}else if(input1.validity.valueMissing){
			// 表示为空
			console.log("input 元素为空.");
		}else if(input1.validity.patternMismatch){
			// 表示正则不匹配
			console.log("正则不匹配.");
		}else if(input1.validity.tooLong){
			console.log("输入内容过长.");
		}
		var e = document.getElementById("e");
		if(e.validity.typeMismatch){
			// 表示类型不匹配
			console.log("email 输入有误.");
		}
		var n = document.getElementById("n");
		if(n.validity.rangeUnderflow){
			// 表示值小于 min 值
			console.log("值过小.");
		}else if(n.validity.stepMismatch){
			console.log("step 不对.");
		}return false;
	}
```

<title>HTML5 的自定义错误提示函数</title>

```
	function formValidate(){
		var i = document.getElementById("i");
		if(i.validity.valueMissing){
			i.setCustomValidity("元素不能为空.");// 为空
		}else if(i.validity.customError){
			// 表示调用 setCustomValidity()方法
			i.setCustomValidity("");
		}
	}	</script>
```

Day02

```
<video autoplay>
        <source src="../DATA/oceans-clip.mp4" />
        <source src="../DATA/oceans-clip.ogv" />
        <source src="../DATA/oceans-clip.webm" />
        不好意思,你的浏览器不支持视频!  </video>
<!-- <video src="../DATA/oceans-clip.mp4" controls></video>
<video src="../DATA/oceans-clip.mp4" autoplay loop></video>-->
    <video src="../DATA/oceans-clip.mp4" controls
                poster="../DATA/oceans-clip.png">     </video>
```

<title>video 元素的事件</title>

```
    <video id="mmedia" src="../DATA/oceans-clip.mp4" controls>
    </video>
    <div id="adv"> <img src="Koala.jpg" width="250px">   </div>
    <script>
        var mmedia = document.getElementById("mmedia");
        mmedia.addEventListener("play",myPlay);
        mmedia.addEventListener("pause",myPause);
        var adv = document.getElementById("adv");
        function myPlay(){
                console.log("视频已播放.");
                adv.style.display = "none"; }
        function myPause(){
                console.log("视频已暂停.");
                adv.style.display = "block"; }
        /*当将<video>元素播放的视频,设置为全屏播放时
            * 显示的图片是无法显示的
            * 将播放视频内容,页面中最顶端的元素 */
```

<title>video 元素的方法和属性</title>

```
<script>
        var btn = document.getElementById("btn");
        btn.addEventListener("click",myClick);
        var mmedia = document.getElementById("mmedia");
        function myClick(){
                if(mmedia.paused){
                        mmedia.play();// 播放视频 - play()
                        btn.value = "暂停";
                }else{
                        mmedia.pause();
                        btn.value = "播放";}     }
```

<title>Canvas 元素</title>

```
    <!--定义<canvas>元素
        * 默认具有大小 300px * 150px
        * <canvas>元素设置 CSS 样式
        * 设置<canvas>元素的高度和宽度
            * style 属性 - 绘制的图形被拉伸
            * width 和 height 属性 - 绘制的图形没有问题 -->
    <canvas id="canvas1"
      style="background:blue;width:100%;height:300px"></canvas>
    <canvas id="canvas2" style="background:pink" width="800px"
height="200px" ></canvas>
    <script>
        // 1. 获取 canvas 元素
        var canvas1 = document.getElementById("canvas1");
        var canvas2 = document.getElementById("canvas2");
        /*2. 创建画布对象
            通过 getContext()方法,创建画布对象
            * 该方法接收一个参数:2d 或 3d
                * 2d - 二维图形
                * 3d - 三维图形
            * 注意
                * 参数的类型为字符串
                * 2d 或 3d 中的"d"必须小写
            * 目前基本上都是 2d 效果 */
        var context1 = canvas1.getContext("2d");
        var context2 = canvas2.getContext("2d");
        // 3. 利用画布对象绘制图形
        context1.fillRect(10,10,100,100);
        context2.fillRect(10,10,100,100);
```

<title>绘制矩形</title>

```
        // 1. 获取<canvas>元素并创建画布对象
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 2. 绘制实心矩形
        context.fillRect(10,10,100,100);
        // 3. 绘制空心矩形
        context.strokeRect(120,10,100,100);
        // 4. 清除指定区域的矩形
        context.fillRect(240,10,100,100);
        context.clearRect(250,20,80,80);
```

<title>设置图形样式</title>

```
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 1. 设置样式
        context.fillStyle = "yellow";
        // 2. 绘制图形
        context.fillRect(10,10,100,100);
        // 再绘制实心矩形(绿色)
        context.fillStyle = "green";
        context.fillRect(10,120,100,100);
        context.strokeStyle = "red";
        context.strokeRect(120,10,100,100);
```

<title>canvas 练习一</title>

```
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        /*需求
            * 设置的颜色 - 随机
                * 单词    * #122313
                * rgb(,,)
```

```
        * 绘制矩形的位置 - 随机
          * 必须要求在 canvas 内部
        * 绘制矩形的大小 - 随机
          * 大小不能大于画布*/
    function paintRect(){
        // 1. 颜色随机
        var r = parseInt(Math.random()*255);
        var g = parseInt(Math.random()*255);
        var b = parseInt(Math.random()*255);
        // 2. 设置样式
        context.strokeStyle = "rgb("+r+","+g+","+b+")";
        // 3. 获取 canvas 的高度和宽度
        const WIDTH = canvas.width;
        const HEIGHT = canvas.height;
        // 4. 位置(x,y)随机
        var x = Math.random()*WIDTH;
        var y = Math.random()*HEIGHT;
        // 5. 大小(width 和 height)随机
        var width = Math.random()*WIDTH;
        var height = Math.random()*HEIGHT;
        // 6. 绘制矩形
        context.strokeRect(x,y,width,height);
    }
    setInterval(paintRect,500);
```

<title>Canvas 线性渐变</title>

```
  <canvas id="canvas" width="500px" height="300px"></canvas>
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    /* 1. 调用 createLinearGradient()方法设置渐变
        * 通过传递四个参数设置基准线的位置
        * 该方法返回线性渐变对象*/
var grad =
  context.createLinearGradient(0,0,canvas.width,canvas.height);
    /*2. 设置渐变的颜色
     渐变对象调用 addColorStop(position,color)方法设置颜色
        * position - 设置当前颜色的位置
          * 值范围为 0-1        * color - 设置的颜色*/
    grad.addColorStop(0,"red");
    grad.addColorStop(0.5,"yellow");
    grad.addColorStop(0.75,"green");
    grad.addColorStop(1,"blue");
    context.fillStyle = grad;// 3. 设置填充样式为线性渐变
    // 4. 绘制矩形
    context.fillRect(0,0,canvas.width,canvas.height);
```

<title>Canvas 射线渐变</title>

```
  <canvas id="canvas" width="500px" height="300px"></canvas>
<script>
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    // 1. 设置射线渐变
var grad = context.createRadialGradient
(canvas.width/2,canvas.height/2,50,canvas.width/2,canvas.height/
2,200);
    grad.addColorStop(0,"red");// 2. 设置渐变颜色
    grad.addColorStop(0.5,"blue");
    grad.addColorStop(1,"green");
    context.fillStyle = grad;// 3. 设置填充样式为渐变
    // 4. 绘制矩形
    context.fillRect(0,0,canvas.width,canvas.height);
```

<title>Canvas 绘制文字</title>

```
  <canvas id="canvas" width="500px" height="400px"></canvas>
  <script>
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.beginPath();// 绘制一条基准线
    context.moveTo(100,0);
    context.lineTo(100,400);
    context.stroke();
    context.font = "bold 48px 微软雅黑";// 设置字体
    context.fillStyle = "blue";     // 设置填充样式
    context.textAlign = "left";// 设置水平对齐
    context.fillText("达内",100,50);// 绘制实心文字
    context.textAlign = "center";
    context.fillText("达内",100,150);
    context.textAlign = "right";
    context.fillText("达内",100,250);
  </script>
```

<title>Canvas 设置阴影</title>

```
<body>
  <canvas id="canvas" width="800px" height="500px"></canvas>
  <script>
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.fillStyle = "green";// 设置填充颜色
    context.shadowColor = "red";// 设置阴影效果
    context.shadowOffsetX = 10;
    context.shadowOffsetY = 10;
    context.shadowBlur = 20;
    // 绘制实心矩形
    context.fillRect(50,50,100,100);
    // 设置字体
    context.font = "bold 48px 微软雅黑";
    // 绘制实心文字
    context.fillText("达内",200,200);
  </script>
```

Day03

```
<title>创建路径绘制矩形或圆形</title>
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 1. 标识创建路径
        context.beginPath();
        // 2. 设置矩形
        context.rect(10,10,100,100);
        // 3. 标识结束
        context.closePath();
        // 4. 绘制实心
        context.fill();
        // 绘制空心矩形
        context.beginPath();
        context.rect(10,120,100,100);
        context.closePath();
        context.stroke();
        // 绘制实心圆形
        context.beginPath();
        context.arc(170,60,50,0,Math.PI*2);
        context.closePath();
        context.fill();
        // 绘制空心圆形
        context.beginPath();
        context.arc(170,170,50,0,Math.PI*2);
        context.closePath();
        context.stroke();
        // 绘制实心弧形
        context.beginPath();
        context.arc(280,60,50,0,Math.PI*3/2,false);
        context.closePath();
        context.fill();
        // 绘制空心弧形
        context.beginPath();
        context.arc(280,170,50,0,Math.PI*3/2);
        context.closePath();
        context.stroke();
    </script>
```

<title>绘制圆形练习</title>

```
// 1. 绘制空心圆形
        context.beginPath();
        context.arc(200,200,100,0,Math.PI*2);
        context.closePath();
        context.stroke();
        // 2. 绘制实心半圆
        context.beginPath();
        context.arc(200,200,100,Math.PI/2,Math.PI*3/2);
        context.closePath();
        context.fill();
        // 3. 绘制黑色圆形
        context.beginPath();
        context.arc(200,250,50,0,Math.PI*2);
        context.closePath();
        context.fill();
        // 4. 绘制白色圆形
        context.fillStyle = "white";
        context.beginPath();
        context.arc(200,150,50,0,Math.PI*2);
        context.closePath();
        context.fill();
        // 5. 绘制一黑一白小圆形
        context.beginPath();
        context.arc(200,250,20,0,Math.PI*2);
        context.closePath();
        context.fill();
        context.fillStyle = "black";
        context.beginPath();
        context.arc(200,150,20,0,Math.PI*2);
        context.closePath();
        context.fill();
    </script>
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 绘制直线
        context.beginPath();
        context.moveTo(10,10);
        context.lineTo(10,200);
        context.closePath();
        context.stroke();
    </script>
```

<title>绘制折线</title>

```
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        context.beginPath();
        context.moveTo(100,10);
        context.lineTo(200,10);
        context.lineTo(200,200);
        context.lineTo(400,200);
        context.closePath();
        context.stroke();
```

<title>绘制折线练习</title>

```
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        context.beginPath();
        context.moveTo(100,10);
```

```
        context.lineTo(100,350);
        context.lineTo(400,350);
        context.stroke();
        for(var i=1;i<10;i++){
            context.beginPath();
            context.moveTo(100+30*i,350);
            context.lineTo(100+30*i,360);
            context.stroke();
        }
        context.font = "18px 微软雅黑";
        context.textAlign = "right";
        context.textBaseline = "top";
        context.fillText("0",100,350);
    </script>
```

<title>设置线条</title>

```
    <canvas id="canvas" width="500px" height="600px"></canvas>
    <script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 设置线的宽度
        context.lineWidth = 10;
        // 设置线的端点形状
        context.lineCap = "round";
        // 设置线的焦点形状
        context.lineJoin = "miter";
        context.miterLimit = 30;
        context.beginPath();
        context.moveTo(100,10);
        context.lineTo(150,300);
        context.lineTo(130,50);
        context.stroke();
    </script>
```

<title>Canvas 绘制图片</title>

```
<canvas id="canvas" width="900px" height="600px"></canvas>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 通过 JavaScript 代码读取图片
        var img = new Image();
        img.src = "Penguins.jpg";
/* 通过 Canvas 的 drawImage()方法进行绘制
            * 通过 new Image()加载图片内容
            * 通过 drawImage()进行绘制图片
            问题
 * 调用 drawImage()进行绘制时,必须保证图片已经读取完成
            解决
            * 读取的图片绑定 onload 事件
              * 在该事件的处理函数中,进行绘制 */
        img.onload = function(){
            context.drawImage(img,0,0,512,384);
            // 绘制矩形
            context.fillRect(100,100,100,100);
        }
```

<title>Canvas 平铺图片</title>

```
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 平铺图片
        // 1. 读取图片
        var img = new Image();
        img.src = "ppp.jpg";
        // 2. 绑定 onload 事件
        img.onload = function(){
            /*3. 平铺图片
              * createPattern(img,type)
                * 作用 - 设置平铺方式
                * 该方法返回平铺对象
              * 设置填充样式为平铺
              * 绘制图形 */
            var ptn = context.createPattern(img,"repeat-y");
            context.fillStyle = ptn;
            context.fillRect(0,0,canvas.width,canvas.height);
        }
    </script>
```

<title>Canvas 切割图片</title>

```
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 1. 绘制图片
        var img = new Image();
        img.src = "Penguins.jpg";
        img.onload = function(){
            context.drawImage(img,0,0,512,384);
        }
        /*2. 切割图片
            创建路径
            * beginPath()
            * rect()或 arc()
            * closePath()
            * clip() - 切割方法*/
        context.beginPath();
        context.arc(350,50,50,0,Math.PI*2);
        context.closePath();
        context.clip();
    </script>
```

<title>Canvas 缩放方法</title>

```
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
```

```
        // 绘制实心矩形
        context.fillStyle = "green";
        context.fillRect(20,20,100,100);
        // 绘制实心矩形(放大)
        context.scale(1.5,1.5);
        context.fillRect(180,20,100,100);
        // 绘制实心矩形(缩小)
        context.scale(0.5,0.5);
        context.fillRect(780,20,100,100);
    </script>
<title>Canvas 平移方法</title>
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        context.fillRect(0,0,100,100);
        context.translate(100,100);
        context.fillRect(0,0,100,100);
        context.translate(100,100);
        context.fillRect(0,0,100,100);
    </script>
<title>Canvas 旋转画布方法</title>
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        // 旋转画布
        context.rotate(30*Math.PI/180);
context.fillRect(canvas.width/2-50,canvas.height/2-50,100,100);
    </script>
<title>画布中心旋转</title>
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        /* 1. 将画布的左上角,平移到画布的中心点
            translate()平移方法 - 相对平移
            * 如果画布中没有任何定位时,相对于画布的左上角*/
        context.translate(canvas.width/2,canvas.height/2);
        function myRotate(){
        context.clearRect(-canvas.width/2,-canvas.height/2,canvas.w
idth,canvas.height);
                // 旋转画布
                context.rotate(10*Math.PI/180);
                // 2. 绘制图形
                context.fillRect(-50,-50,100,100);
        }
        setInterval(myRotate,100);
    </script>
<title>螺旋图练习</title>
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        var btn = document.getElementById("btn");
        btn.onclick = function(){
                // 初始化的第一个矩形
                context.translate(150,30);
                context.scale(0.95,0.95);
                context.fillStyle = "pink";
                context.globalAlpha = 0.5;
                context.fillRect(0,0,100,50);
                for(var i=0;i<50;i++){
                        // 上述设置重新绘制
                        context.translate(25,10);
                        context.scale(0.95,0.95);
                        context.rotate(Math.PI/15);
                        context.fillRect(0,0,100,50);
                }
        }
    </script>
<title>Chart.js 绘制饼状图</title>
<script>
        // 3. 获取<canvas>元素,并且创建画布对象
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        /* 4. 创建 Chart 对象
            new Chart(context)构造器 */
        var chart = new Chart(context);
        // 配置全局选项
        Chart.defaults.global.animation = false;
        //Chart.defaults.global.responsive = true;
        // 5. 设置相关数据
        var data = [
                {   // 设置饼状图的值
                        value : 50,
                        // 设置饼状图的颜色
                        color : "pink",
                        // 设置饼状图鼠标悬停的颜色
                        highlight : "deeppink",
                        // 设置饼状图文字提示内容
                        label : "Pink"
                },{
                        value : 350,
                        color : "green",
                        highlight : "lightgreen",
                        label : "Green"
                },{
                        value : 450,
                        color : "blue",
                        highlight : "lightblue",
                        label : "Blue"
                }
```

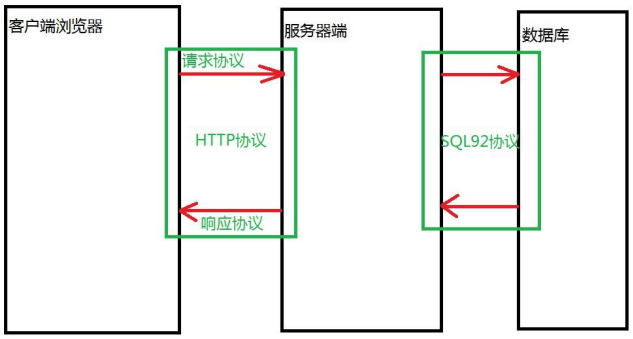
```
        ];
        // 6. 进行绘制
        chart.Pie(data);
    </script>
```

<title>Chart.js 绘制柱状图</title>

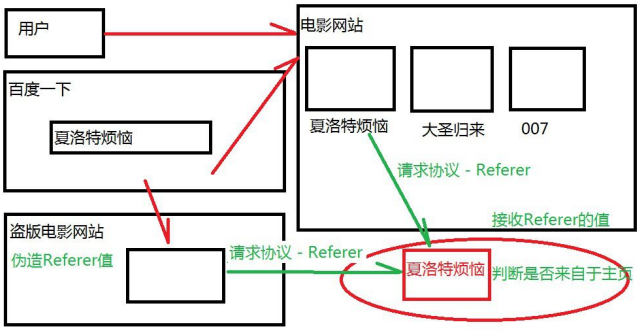
```
<script>
        var canvas = document.getElementById("canvas");
        var context = canvas.getContext("2d");
        var chart = new Chart(context);
        var data = {
            // 柱状图的提示内容
labels: ["一月", "二月", "三月", "四月", "五月", "六月", "七月"],
            datasets: [
                {       // 文字提示内容
                    label: "My First dataset",
                    // 设置填充颜色
                    fillColor: "rgba(220,220,220,0.5)",
                    // 设置描边颜色
                    strokeColor: "rgba(220,220,220,0.8)",
                    // 设置鼠标悬停填充颜色
                    highlightFill: "rgba(220,220,220,0.75)",
                    // 设置鼠标悬停描边颜色
                    highlightStroke: "rgba(220,220,220,1)",
                    // 设置柱状图的数据
                    data: [65, 59, 80, 81, 56, 55, 40]
                },{
                    label: "My Second dataset",
                    fillColor: "rgba(151,187,205,0.5)",
                    strokeColor: "rgba(151,187,205,0.8)",
                    highlightFill: "rgba(151,187,205,0.75)",
                    highlightStroke: "rgba(151,187,205,1)",
                    data: [28, 48, 40, 19, 86, 27, 90]
                }
            ]
        };
        chart.Bar(data);
    </script>
```

Cookie 在 Web 应用的作用

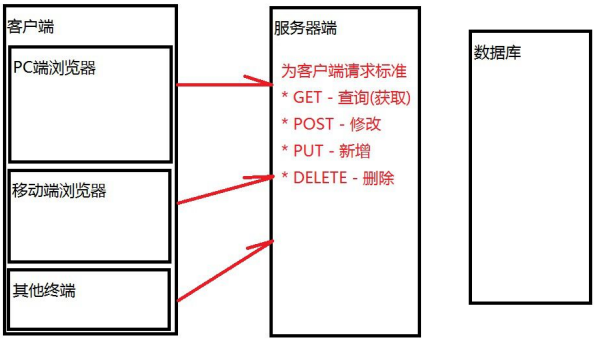
HTTP 协议的含义及作用



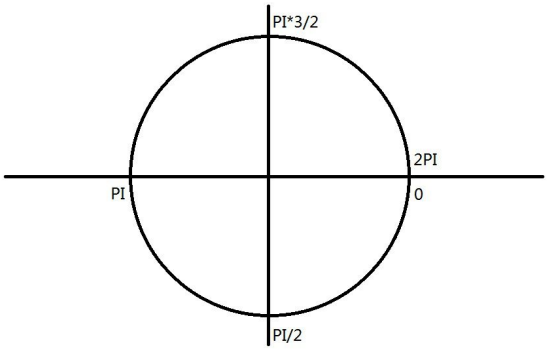
Referer 防盗链流程图



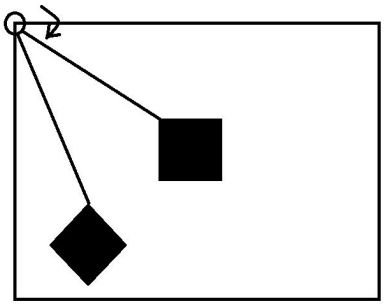
标准式 API



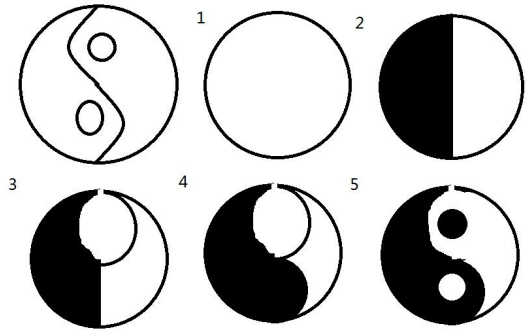
创建路径绘制圆形的机制



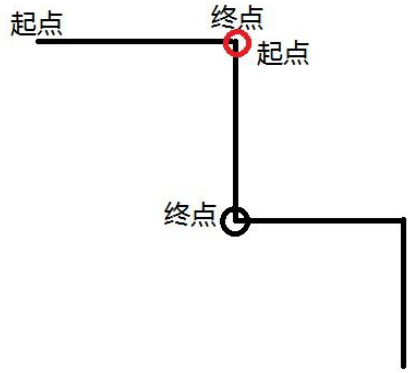
画布旋转方式分析图



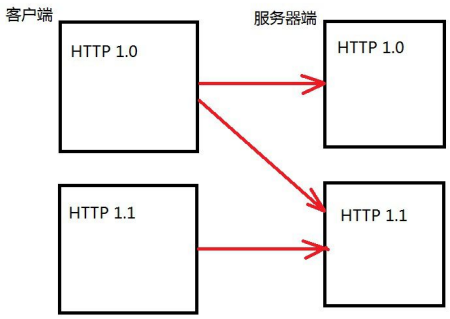
绘制太极图分析



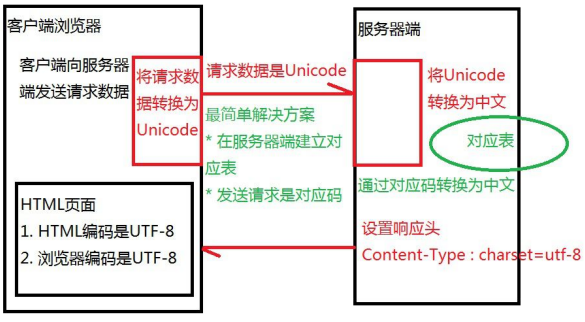
绘制折线的原理分析图



客户端与服务器端的 HTTP 版本对应图



客户端与服务器端交互的中文乱码问题



Day04

```
<!-- 将 SVG 文件当做普通的图片引入 -->
  <img src="01_svg.svg">
 <!--HTML5 之后,将 SVG 的标签直接定义在 HTML 页面中
      * <svg></svg>元素 - 表示该标签内的是 SVG 图形
        * width 和 height 属性 - 设置 SVG 的宽度和高度
        * style 属性 - 使用 CSS 中的属性进行设置-->
  <svg style="background:pink;width:500px;height:500px">
      <rect x="100" y="100" width="100" height="100" />
  </svg>
```

<title>SVG 绘制矩形</title>

```
  <svg width="500px" height="500px">
      <rect x="10" y="10" width="100" height="100"
            fill="white" stroke="black" stroke-width="5" />
      <rect x="110" y="110" width="100" height="100"
            style="fill:lightgreen;stroke:red;stroke-width:5;" />
      <rect x="210" y="210" width="100" height="100" rx="5"
ry="5" fill-opacity="0.5" />
  </svg>
 </body>
```

<title>SVG 绘制圆形</title>

```
  <svg width="500px" height="500px">
      <circle cx="100" cy="100" r="100" fill="lightgreen" />
<circle cx="200" cy="200" r="100" fill="white" stroke="black" />
  </svg>
```

<title>SVG 绘制椭圆</title>

```
  <svg width="500px" height="500px">
      <ellipse cx="200" cy="200" rx="100" ry="100" />
  </svg>
```

<title>SVG 绘制直线</title>

```
  <svg width="500px" height="500px">
      <line x1="100" y1="100" x2="400" y2="100" stroke="black"
stroke-width="10" />
  </svg>
```

<title>SVG 绘制折线</title>

```
  <svg width="500px" height="500px">
      <rect x="100" y="100" width="50" height="50" />
      <polyline points="10,10 200,10 200,200 10,200 10,5"
stroke="black" fill-opacity="0" stroke-width="10" />
  </svg>
```

<title>SVG 绘制多边形</title>

```
  <svg width="500px" height="500px">
      <!--
      <polygon points="10,10 200,10 200,200 10,200 110,110
10,10" />
      -->
      <polygon points="60,20 100,40 100,80 60,100 20,80 20,40"
/>
  </svg>
```

<title>将绘制图形进行分组</title>

```
<div id="show"></div>
<script>
    var div = document.getElementById("show");
    var two = new Two({
        width : 500,
        height : 500
    });
    two.appendTo(div);
    // 设置图形
    var circle = two.makeCircle(72, 100, 50);
    var rect = two.makeRectangle(213, 100, 100, 100);
    // 进行分组 - 该方法返回分组对象
    var group = two.makeGroup(circle,rect);
    // 针对这一组设置样式
    group.fill = "#FF8000";
    group.stroke = "orangered";
    group.linewidth = 5;
    // 设置矩形的样式
    rect.fill = "rgba(0, 200, 255, 0.75)";
    rect.stroke = "#1C75BC";
    // 绘制图形
    two.update();
```

<title>SVG 绘制线型渐变</title>

```
<svg width="500px" height="500px">
    <!--* 先定义<defs>元素,渐变元素定义该元素内
            * 定义<defs>元素的内容 - 允许重复使用
          * 定义<defs>元素内的元素,不会直接显示在页面中
            * <defs>元素是起始标签
        * 再定义<linearGradient>线性渐变元素
            * <linearGradient>元素是起始标签
            * 线性渐变的基准线:起点(x1,y1)和终点(x2,y2)
            * 定义 id 属性值 - 绘制图形时,赋值给 fill 属性
        * 设置渐变的颜色,使用<stop>元素
            * offset - 设置渐变颜色的位置
            * stop-color - 设置渐变的颜色
        * 绘制图形时,使用 fill 属性设置渐变样式
            fill="url(#渐变元素的 id 属性值)"
        * 注意 - 以下设置的值的单位为百分值
            * 设置基准线的起点和终点坐标值
            * 设置渐变颜色的位置-->
    <defs>
    <linearGradient id="grad" x1="0%" y1="0%" x2="100%"
y2="100%">
            <stop offset="0%" stop-color="red" />
            <stop offset="50%" stop-color="yellow" />
            <stop offset="100%" stop-color="blue" />
        </linearGradient></defs>
<rect x="0" y="0" width="500" height="500"
                fill="url(#grad)" >                </svg>
```

<title>SVG 绘制射线渐变</title>

```
<svg width="500px" height="500px">
    <defs>
        <!--<radialGradient>射线渐变元素
            * fx 和 fy - 设置起点基准圆的圆心(没有半径值)
            * cx 和 cy - 设置终点基准圆的圆心
            * r - 设置终点基准圆的半径
            注意
            * 都使用百分值,最大值为 100%-->
        <radialGradient id="grad" fx="-10%" fy="-10%"
cx="110%" cy="110%" r="50%">
            <stop offset="0%" stop-color="red" />
            <stop offset="50%" stop-color="yellow" />
            <stop offset="100%" stop-color="blue" />
        </radialGradient>
    </defs>
<rect x="0" y="0" width="500" height="500" fill="url(#grad)" />
</svg>
```

<title>SVG 绘制高斯模糊</title>

```
<svg width="500px" height="500px">
    <!-- 设置滤镜 - 可重复使用 -->
    <defs>
        <!-- 定义滤镜元素 - <filter>元素
            * 需要定义 id 属性值
              * 绘制图形时,将其设置为图形的样式
            * 绘制图形时
              * 使用 filter 属性设置滤镜效果
                filter="url(#滤镜元素的 id)"-->
        <filter id="myfilter">
        <!-- 定义高斯模糊元素 - <feGaussianBlur>元素
                * in="SourceGraphic" - 固定写法
            * stdDeviation - (Number)设置高斯模糊的程度
                  * 值越大,模糊程度越大-->
    <feGaussianBlur in="SourceGraphic" stdDeviation="5" />
        </filter>
    </defs>
    <rect x="10" y="100" width="100" height="100" fill="green"
filter="url(#myfilter)" />
<rect x="120" y="100" width="100" height="100" fill="green" />
</svg>
```

<title>SVG 练习</title>

```
<svg width="500px" height="500px">
    <defs>
<linearGradient id="grad" x1="0%" y1="0%" x2="100%" y2="0%">
            <stop offset="0%" stop-color="red" />
            <stop offset="100%" stop-color="blue" />
        </linearGradient>
        <filter id="filter">
<feGaussianBlur in="SourceGraphic" stdDeviation="5" />
        </filter>
```

```
        </defs>
<rect x="100" y="100" width="100" height="100" fill="url(#grad)"
filter="url(#filter)" />
    </svg>
  </body>
```

<title>如何使用 Two.js 库</title>

```
    <!-- 1. 引入 Two.js 库文件 -->
    <script src="two.js"></script>
    <!-- 2. 定义用于显示图形的容器 -->
    <div id="mydiv"></div>
    <script>
        // 3. 获取 HTML 页面的容器元素
        var div = document.getElementById("mydiv");
        // 4. 设置 Two 对象的相关配置
        var params = {
                width : 500,        // 设置 SVG 的宽度
                height : 500        // 设置 SVG 的高度
        };
        // 5. 创建 Two 对象
        var two = new Two(params);
        // 6. 将创建的 Two 对象,添加到 HTML 页面的容器中
        two.appendTo(div);
    </script>
```

<title>Two.js 库绘制静态图形</title>

```
    <div id="show"></div>
    <script>
        var div = document.getElementById("show");
        var two = new Two({
                //type : Two.Types.canvas,
                width : 500,
                height : 500
        });
        two.appendTo(div);
        //  如何绘制图形
        // a. 设置图形
        two.makeLine(10,10,300,10);
        var circle = two.makeCircle(200,200,30);
        var ellipse = two.makeEllipse(100,100,100,50);
        // b. 设置样式
        circle.fill = "#FF8000";
        circle.stroke = "orangered";
        circle.linewidth = 5;
        ellipse.fill = "rgba(0, 200, 255, 0.75)";
        ellipse.stroke = "#1C75BC";
        ellipse.linewidth = 5;
        // c. 绘制(更新)图形
        two.update();
    </script>
```

Day05

<title>地图展示</title>

```
        <!-- 1. 引入百度地图的 JS 文件
                * 在线文件(可以联网状态下)
                * 地址
                http://api.map.baidu.com/api?v=2.0&ak=您的密钥-->
        <script
src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbO
nXkztdgSQlQ"></script>
</head>
<body>
        <!-- 2. 定义用于显示百度地图的容器元素 -->
        <div id="allmap"></div>
</body>
<script type="text/javascript">
        /*3. 创建一个百度地图的 Map 对象
                new BMap.Map(元素的 ID) */
        var map = new BMap.Map("allmap");
        /* 4. 设置百度地图显示的中心和级别
                centerAndZoom(center,zoom)
                 * center - (String 类型),设置地图显示的中心
                 * zoom -  设置地图显示的级别(3 - 19) */
        map.centerAndZoom("北京", 11);
</script>
```

<title>地图的控件展示</title>

```
    <script
src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbO
nXkztdgSQlQ"></script>
  </head>
  <body>
    <div id="allmap" style="width:500px;height:500px;"></div>
    <script>
        var map = new BMap.Map("allmap");
        map.centerAndZoom("北京",12);
        /*向地图添加比例尺
            *  创建一个比例尺对象
                new BMap.ScaleControl(options)
                 * anchor -  设置控件在地图中显示的位置
                    * BMAP_ANCHOR_TOP_LEFT -  左上角
                    * BMAP_ANCHOR_BOTTOM_LEFT -  左下角
                    * BMAP_ANCHOR_TOP_RIGHT -  右上角
                    * BMAP_ANCHOR_BOTTOM_RIGHT -  右下角
            *  通过 Map 对象添加控件
                addControl(比例尺对象);*/
        var scale = new BMap.ScaleControl({
                anchor : BMAP_ANCHOR_BOTTOM_RIGHT
        });
        map.addControl(scale);
```

```
/*向地图添加缩放平移控件
        * 创建缩放平移对象
          new BMap.NavigationControl(options)
           * anchor - 设置控件在地图中显示的位置
             * BMAP_ANCHOR_TOP_LEFT - 左上角
             * BMAP_ANCHOR_BOTTOM_LEFT - 左下角
             * BMAP_ANCHOR_TOP_RIGHT - 右上角
             * BMAP_ANCHOR_BOTTOM_RIGHT - 右下角
          * type - 平移和缩放控件样式
* BMAP_NAVIGATION_CONTROL_SMALL：仅包含平移和缩放按钮
* BMAP_NAVIGATION_CONTROL_PAN:仅包含平移按钮
 * BMAP_NAVIGATION_CONTROL_ZOOM：仅包含缩放按钮
        * 通过 Map 对象添加控件
          addControl(比例尺对象); */
      var nav = new BMap.NavigationControl({
          anchor : BMAP_ANCHOR_TOP_RIGHT,
          type : BMAP_NAVIGATION_CONTROL_SMALL
      });
      map.addControl(nav);
      /*向百度地图添加地图类型
        * 创建地图类型对象
          new BMap.MapTypeControl()
        * 通过 Map 对象添加控件
          addControl(比例尺对象);*/
      var type = new BMap.MapTypeControl({
          anchor : BMAP_ANCHOR_TOP_LEFT
      });
      map.addControl(type);
  </script>
 </body>
<title>地图的覆盖物展示</title>
  <script
src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>
 </head>
 <body>
  <div id="allmap" style="width:500px;height:500px;"></div>
  <script>
      var map = new BMap.Map("allmap");
      map.centerAndZoom("北京",14);
      /* 向地图添加标注
        * 创建一个地理点的坐标值
          new BMap.Point(lng,lat)
           * lng - 表示经度
           * lat - 表示纬度
        * 创建一个标注对象
          new BMap.Marker(point)
           * point - 标注所在的坐标点
        * 通过 Map 对象添加覆盖物
          addOverlay(overlay)*/
      var point = new BMap.Point(116.404, 39.915);
      var marker = new BMap.Marker(point);
      map.addOverlay(marker);
      /*向地图添加信息窗口(点击标注,打开信息窗口)
        * 设置信息窗口相关配置
          {
              width : 设置信息窗口的宽度,
              height : 设置信息窗口的高度,
              title : 设置信息窗口的标题,
              enableMessage : 设置信息是否允许发送短信,
              message : 设置信息窗口的内容
          }
        * 创建信息窗口对象
          new BMap.InfoWindow(addr,opts)
           * addr - 设置当前标注坐标点的地址
           * opts - 设置当前信息窗口的配置
        * 通过 Map 对象打开信息窗口
          openInfoWindow(info,point)
           * info - 信息窗口对象
           * point - 坐标点对象*/
      var opts = {
          width : 200,
          height : 100,
          title : "北京天安门"
      }
      var info = new BMap.InfoWindow("地址:北京市长安街 0 号
",opts);
      marker.addEventListener("click",function(){
          map.openInfoWindow(info,point);
      });
  </script>
 </body>
<title>地图右键菜单展示</title>
  <script
src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbOnXkztdgSQlQ"></script>
 </head>
 <body>
  <div id="allmap" style="width:500px;height:500px;"></div>
  <script>
      var map = new BMap.Map("allmap");
      map.centerAndZoom("北京");
      var point = new BMap.Point(116.404, 39.915);
      /*如何自定义地图的鼠标右键菜单
        * 创建自定义右键菜单对象
          new BMap.ContextMenu()
        * 设置自定义右键菜单的选项
          [
              {
                  text : 菜单名称,
```

```
                callback : 点击菜单的回调函数
            }
        ]
    * 向自定义右键菜单添加菜单选项
        * 创建菜单选项对象
            new BMap.MenuItem(选项)
        * 自定义右键菜单对象
            addItem(menuitem)
    * 通过 Map 对象添加右键菜单
        addContextMenu(menu)*/
    var menu = new BMap.ContextMenu();
    var menuItems = [
        {
            text:'放大',
            callback:function(){map.zoomIn()}
        },{
            text:'缩小',
            callback:function(){map.zoomOut()}
        }
    ];
    for(var i=0;i<menuItems.length;i++){
var menuItem = new BMap.MenuItem
                (menuItems[i].text,menuItems[i].callback,100);
        menu.addItem(menuItem);
    }
    map.addContextMenu(menu);
  </script>
 </body>
```

<title>地图地址解析展示</title>

```
  <meta charset="utf-8" />
  <script
src="http://api.map.baidu.com/api?v=2.0&ak=HbUVYMUg6PwbO
nXkztdgSQlQ"></script>
 </head>
 <body>
  <div id="allmap" style="width:500px;height:500px;"></div>
  <script>
    var map = new BMap.Map("allmap");
    //map.centerAndZoom("北京",12);
    /*完成地址解析
        * 创建地址解析器对象
            new Geocoder()
        * 调用 getPoint(addr,callback,city)方法
            * addr - 要解析的地址
             * callback - 回调函数
                function(point){}
                * 如果对地址的定位解析成功的话
                    * 在回调函数中包含一个 Point 参数
                * 如果对地址的定位解析失败的话
                    * 返回一个 Null 值
                * city - 地址所在的城市*/
    var geocoder = new BMap.Geocoder();
    geocoder.getPoint("北京市海淀区万寿路西街 2 号
",function(point){
            // 使用坐标点作为中心点,重新设置中心
            map.centerAndZoom(point,18);
            //map.setCenter(point);
            var marker = new BMap.Marker(point);
            map.addOverlay(marker);
    },"北京市");
  </script>
 </body>
```

<title>源元素事件</title>

```
 <div>
    <img id="img" width="256" src="Koala.jpg">
 </div>
 <script>
    // 获取源元素
    var img = document.getElementById("img");
    // 为源元素绑定事件
    // (只被触发一次)开始拖放时
    img.addEventListener("dragstart",myDragstart);
    // (重复被触发)拖放一直被触发
    img.addEventListener("drag",myDrag);
    // (只被触发一次)结束拖放时
    img.addEventListener("dragend",myDragend);
    // 定义事件的处理函数
    function myDragstart(event){
    console.log("开始拖放: "+event.pageX+", "+event.pageY);
    }
    function myDrag(event){
    console.log("正在拖放: "+event.pageX+", "+event.pageY);
    }
    function myDragend(event){
    console.log("结束拖放: "+event.pageX+", "+event.pageY);
    }
 </script>
```

<title>目标元素事件</title>

```
 <!-- 作为拖放的源元素 -->
 <div id="d1">
    <img id="img" width="256" src="Koala.jpg">
 </div>
 <!-- 作为拖放的目标元素 -->
 <div id="d2"></div>
 <script>
    // 获取目标元素
    var d2 = document.getElementById("d2");
    // 为目标元素绑定事件
    d2.addEventListener("dragenter",myDragenter);
    d2.addEventListener("dragover",myDragover);
```

```
    d2.addEventListener("drop",myDrop);
    d2.addEventListener("dragleave",myDragleave);
    // 定义事件的处理函数
    function myDragenter(event){
        // 该方法用于阻止默认行为
        //event.preventDefault();
        console.log("大爷来啦...");
    }
    function myDragover(event){
        // 该方法用于阻止默认行为
        event.preventDefault();
        console.log("大爷又来啦...");
    }
    function myDrop(event){
        // 该方法用于阻止默认行为
        //event.preventDefault();
        console.log("大爷别走了...");
    }
    function myDragleave(event){
        // 该方法用于阻止默认行为
        //event.preventDefault();
        console.log("大爷下次再来啊...");
    }
</script>
```

<title>HTML5 实现拖放效果</title>

```
<!-- 作为拖放的源元素 -->
<div id="d1">
    <img id="img" width="256" src="Koala.jpg">
</div>
<!-- 作为拖放的目标元素 -->
<div id="d2"></div>
<script>
    // 1. 获取源元素和目标元素
    var img = document.getElementById("img");
    var d2 = document.getElementById("d2");
    /*2. 为源元素和目标元素绑定必要事件
        * 源元素 - 绑定 dragstart 事件
        * 目标元素 - 绑定 dragover 和 drop 事件 */
    img.addEventListener("dragstart",myDragstart);
    d2.addEventListener("dragover",myDragover);
    d2.addEventListener("drop",myDrop);
    // 3. 定义事件的处理函数
    function myDragstart(event){
        // 将源元素的关键数据,保存到 dataTransfer 对象中
        var data = img.src;
        event.dataTransfer.setData("text",data);
    }
    function myDragover(event){
        // 用于阻止 HTML 页面的默认行为
        event.preventDefault();
    }
    function myDrop(event){
        // 将源元素的关键数据得到,实现拖放效果
        var data = event.dataTransfer.getData("text");
        var myImg = document.createElement("img");
        myImg.setAttribute("src",data);
        myImg.setAttribute("width","256");
        d2.appendChild(myImg);
    }
</script>
```

<title>HTML5 页面外实现拖放</title>

```
<!-- div 元素为目标元素 -->
<div id="d"></div>
<script>
    // 1. 获取目标元素
    var d = document.getElementById("d");
    // 2. 为目标元素绑定事件
    d.addEventListener("dragover",myDragover);
    d.addEventListener("drop",myDrop);
    // 3. 定义事件的处理函数
    function myDragover(event){
        event.preventDefault();
        var filelist = event.dataTransfer.files;
        for(var i=0;i<filelist.length;i++){
            console.log(filelist[i]);
        }
    }
    function myDrop(event){
        console.log("drop: "+event);
    }
</script>
```

Day06

<title>JS 完成计时器</title>

```
<body>
  <input type="button" id="start" value="开始">
  <input type="button" id="stop" value="结束">
  <div id="showtime"></div>
  <script>
      // 1. 获取开始和结束按钮
      var start = document.getElementById("start");
      var stop = document.getElementById("stop");
      // 3. 获取 div 元素
      var showtime = document.getElementById("showtime");
      // 4. 定义用于计数的变量
      var num = 0;
      // 2. 为按钮绑定事件
      start.onclick = startAdd;
      function startAdd(){
            // 开始计数
            showtime.innerHTML = num;
            num++;
            t = setTimeout(startAdd,1000);
      }
      stop.onclick = function(){
            // 停止计数
            clearTimeout(t);
      }
  </script>
</body>
```

<title>Worker 完成计时器</title>

```
  <input type="button" id="start" value="开始">
  <input type="button" id="stop" value="结束">
  <div id="showtime"></div>
  <script>
      // 获取开始和结束按钮
      var start = document.getElementById("start");
      var stop = document.getElementById("stop");
      var showtime = document.getElementById("showtime");
      var worker;
      // 为按钮绑定 onclick 事件
      start.onclick = function(){
            // 1. 检测浏览器是否支持 Worker
            if(typeof(Worker) !== "undefined"){
                  /* 2. 创建 Worker 对象
                        * 注意 - 指定对应 Worker 文件
                         var worker = new Worker(url);
                         * url - Worker 文件的路径*/
                  worker = new Worker("worker1.js");
                  /*3. 绑定 onmessage 事件
                        * 注意
                        * 将 onmessage 事件绑定在 Worker 对象
                         * 触发
                        * Worker 文件调用 postMessage()方法时
                        * 处理函数
      * 具有形参允许接收 postMessage()方法传递的数据内容 */
                        worker.onmessage = function(event){
                              showtime.innerHTML = event.data;
                        }
                  }else{// 不支持
                        showtime.innerHTML = "<h3>该浏览器不支持
Web Worker.</h3>";
                  }
      }
      stop.onclick = function(){
            // 终止与 Worker 的通信 - terminate()
            worker.terminate();
      }
  </script>
```

<title>与 Worker 双向通信</title>

```
  <input type="text" id="data">
  <input type="button" id="btn" value="发送">
  <br>
  <div id="showme"></div>
  <script>
      // 获取 div 元素
      var showme = document.getElementById("showme");
      // 定义 Worker 对象
      var worker;
      // 1. 检测是否支持 Worker
      if(typeof(Worker) !== "undefined"){
            // 2. 创建 Worker 对象
            worker = new Worker("worker2.js");
            // 3. 向 Worker 文件传递消息
            worker.postMessage("init");
            // 4. 绑定 onmessage 事件
            worker.onmessage = myMessage;
      }else{// 不支持
            showme.innerHTML = "对不起,你的浏览器不支持.";
      }
      // 定义处理函数
      function myMessage(event){
            showme.innerHTML += event.data+"<br>";
      }
      var data = document.getElementById("data");
      var btn = document.getElementById("btn");
      btn.onclick = function(){
            var value = data.value;
            worker.postMessage(value);
      }
  </script>
```

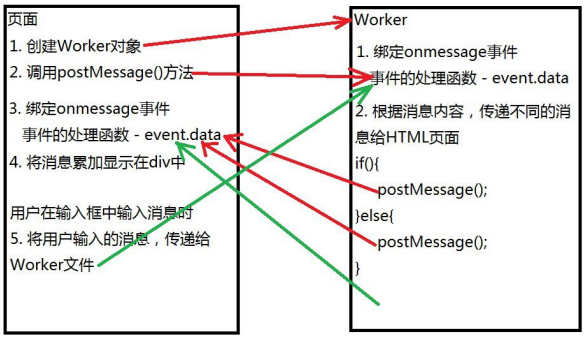
```
<title>session Storage</title>
  <script>
    // 测试 session Storage 的 API
    /*1. 如何获取 session Storage
        * 通过 window 对象的 sessionStorage 属性
        * 返回的是(session)Storage 对象
        注意
        * 使用 window 对象的属性和方法时,可省略"window."
        * sessionStorage 对象可直接得到*/
    var ss = window.sessionStorage;
    console.log(ss);
    /*2. setItem(key,value)
        * 如何生成 key
          * 时间戳 - new Date().getTime();
        * 如果 key 存在
          * setItem(key,value) - 修改指定数据内容
        * 如果 key 不存在
          * setItem(key,value) - 新增指定数据内容*/
    // 时间戳
    var key = new Date().getTime();// 精确到毫秒
    sessionStorage.setItem("1450075181102","zhouzhiruo");
    /*3. getItem(key)
        * key - 存在的 Storage 的数据的标识
        注意
        * 获取数据,必须已知 key 值*/
    var value = sessionStorage.getItem("1450075181102");
    document.body.innerHTML = value;
    /*4. removeItem(key)
        * key - 存在的 Storage 的数据的标识*/
    sessionStorage.removeItem("1450075181102");
    // 5. length - 获取当前存储数据的个数
    console.log(sessionStorage.length);
    /*6. key(index)
        * 作用 - 根据索引值得到 key 值*/
    for(var i=0;i<sessionStorage.length;i++){
        var key = sessionStorage.key(i);
        console.log(key);
    }
    // clear() - 清空
    sessionStorage.clear();
  </script>
</body>
```

```
<title>Web 存储 API 案例</title>
  <h1>Web 版本简单记事本</h1>
  <textarea id="data"></textarea>
  <br><br>
  <input type="button" id="save" value="保存">
  <input type="button" id="read" value="读取">
  <input type="button" id="update" value="修改">
  <input type="button" id="del" value="删除">
  <br><br>
  <!-- 隐藏域 -->
  <input type="hidden" id="mykey">
  <div id="datas"></div>
  <script>
    var mykey = document.getElementById("mykey");
    /*需求 - localStorage
        * 保存 - 用户输入完毕后,点击该按钮进行保存
          * 要求 - 必须输入内容
        * 读取 - 将已存储的所有数据读取,并且以<table>表格格式显示在 HTML 页面中
        * 修改 - 选择<table>表格某一条数据,进行修改
        * 删除 - 选择<table>表格某一条数据,进行删除 */
    // 1. 保存功能
    var save = document.getElementById("save");
    var data = document.getElementById("data");
    save.onclick = function(){
        if(data.value == ""||data.value == null){
            alert("请先输入内容.");
        }else{
            var key = mykey.value;
            if(key == "" || key == null){
                key = new Date().getTime();
            }
            localStorage.setItem(key,data.value);
            data.value = "";
            mykey.value = "";
            alert("输入的内容已保存.");
        }
    }
    // 2. 读取功能
    var read = document.getElementById("read");
    var datas = document.getElementById("datas");
    read.onclick = myRead;

    function myRead(){var table = "<table border='1'>
            <tr><th></th><th>key</th><th>value</th></tr>";
        for(var i=0;i<localStorage.length;i++){
            var key = localStorage.key(i);
            var value = localStorage.getItem(key);
            table += "<tr><td><input type='radio' name='txts' value='"+key+"'></td><td>"+key+"</td><td>"+value+"</td></tr>";
        }
        table += "</table>";
        datas.innerHTML = table;
    }
    // 3. 修改功能
    var update = document.getElementById("update");
    update.onclick = function(){
        var radios = document.getElementsByName("txts");
```

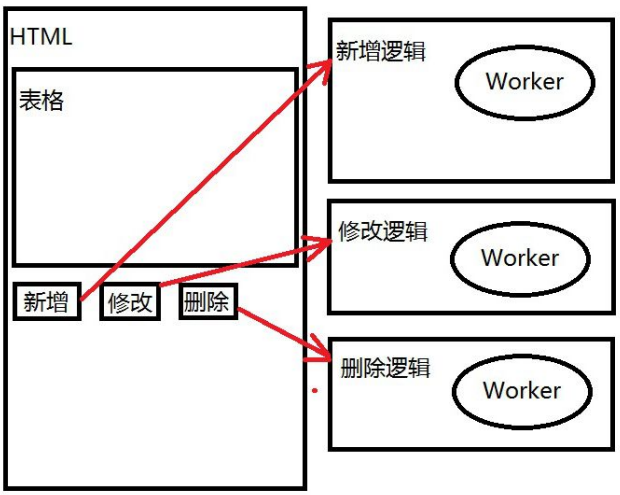
```
            for(var i=0;i<radios.length;i++){
                var radio = radios[i];
                if(radio.checked){
                    var key = radio.value;
                    mykey.value = key;
                    break;
                }
            }
            var value = localStorage.getItem(key);
            data.value = value;
        }
        // 4. 删除功能
        var del = document.getElementById("del");
        del.onclick = function(){
            var radios = document.getElementsByName("txts");
            for(var i=0;i<radios.length;i++){
                var radio = radios[i];
                if(radio.checked){
                    var key = radio.value;
                    break;
                }
            }
            localStorage.removeItem(key);
            myRead();
        }
    </script>
</body>
```

HTML 与 Worker 的双向通信



HTTP 协议的问题解析

WebWorker 的含义



Worker 实现计时器分析图